



richardfrancis.info

Web | Data | Process

Digital Design and Management Services



General Practitioner Cloud Solution [GP-VPC]

Abstract

During the coronavirus pandemic, a GP decides to move his surgery to the cloud in the hope that he will be able to test more patients. He recruits a cloud architect to design and report on the architecture. This report outlines a deep dive into AWS, and considers the potential for a cloud surgery. Looks at numerous services provided by AWS, and documents the processes used to prototype the cloud surgery.

Contents

1. Introduction

2. Requirements Analysis

2.1 Cloud Service Provider

3. Cloud Architecture

3.1 System Walkthrough

3.2 Identity and Access (IAM)

3.2.1 Shared Responsibility

3.3 Virtual Private Cloud (VPC)

3.3.1 Components

3.3.2 VPC Peering

3.4 Elastic Compute Cloud (EC2)

3.5 Relational Database Service

3.6 Monitor and Respond

3.6.1 CloudWatch

3.6.2 Lambda

4. Critical Analysis

5. Conclusion

References

Appendices

A. Cloud Development

1.1 GP-VPC.jpg

1.2 Cloud Overview.jpg

1.3 Console Development

1.4 Cloud Formation

1.4.1 GP-STACK.yaml

1.4.2 GP-STACK.designer

1.4.3 GP-STACK-SET

B. Database Development

2.1 GP-ERD.jpg

2.2 GPDB.sql

C. Website Development

3.1 Website Architecture.jpg

3.2 Patient.php

D. Project Management

4.1 Meeting Minutes

4.2 Documentation

E. Miscellaneous

5.1 CloudTrail Logs

5.2 IAM.json

5.3 VPC Micro Services

5.4 Resources

1 Introduction

A General Practitioner (GP) has decided to move his information system to the cloud. He predicts that with the coronavirus pandemic his surgery will become very busy. He speaks to a team of developers and asked them to come up with a report detailing; feasibility, usability, cost efficiency, security, reliability and performance. The GP would also like to know the reasons for our choice of data center, cloud platform, and services. And finally the GP would like a prototype of the system.

Cloud computing has many advantages over a traditional computing environment. For example, capital expense is replaced with variable expense. This means our GP will no longer have to make large investments in computer hardware. Cloud Service Provider (CSP) economies of scale, which ensures our GP will pay minimum costs for operating in the cloud, a benefit from the many thousands of cloud users the CSP has already registered. Stop guessing capacity, increased speed and agility the list is endless.

This report discusses cloud computing, determines if a GP surgery could be based in the cloud, and looks at the architecture of cloud systems that could meet the needs of the GP. The sections are logically laid out, starting with a requirements analysis and finishing with a conclusion.

2 Requirements Analysis

The system modelled in this report acts as a starting point, a base from which the GP can continue to develop and improve the system and its processes. The purpose of this system is to update and upgrade the GP's old on-premises information system to that of a new elastic cloud based system.

The GP needs to communicate with all his patients, keep confidential records of their health, financial transactions and contact details. He needs to keep records of receptionists and nurses; wages, timekeeping and rota. Finally, he needs to migrate data from an on-premises environment to the cloud.

The key characteristic of this system is the data it holds. It is sensitive, protected by law and extremely valuable. The data must be available to all personnel authorised to access it, there must be no failure, data loss or theft. The system must be cost efficient and perform under extreme conditions. The table below lists the functional and non-functional requirements for the cloud system. This is not an exhaustive list of requirements.

Non-Functional Requirements	Functional Requirements
Compliance	Database
Usability	Compute
Reliability	Web Server
Performance	Lambda Functionality
Supportability	Storage
Cost efficiency	Respond to security threats (automated)
Efficiency	Management (simplified)
Availability	External and Internal Interface
Scalability	Security
Simple to use	Network

An Interface is required where staff can update data, patients can book appointments, and the Doctor can access patient records. The system must be able to respond to security threats and be automated where possible. The system should be simple to manage, the Doctor should not have to spend too much time administering.

Although the GP will not have time to manage the system, there are legal requirements and data protection laws that dictate what type of deployment model is compliant. Infrastructure as a Service (IaaS) is described by Rountree and Castrillo (2013, pp 70) as “*computing power, storage, networking and operating systems*”. IaaS provides the bare bones of a system, meaning that all compliance laws and regulations can be strictly adhered to because the GP is in control.

2.1 Cloud Service Provider (CSP)

The choice of data center relates directly to the choice of service provider. Amazon Web Services (AWS) is by far the largest service provider with global infrastructure worth billions. The services they provide are well documented and user friendly. The well architected framework, discussed in the next section, provides a set of questions and guides on how to best implement a cloud system. The framework and our non-functional requirements are synchronised.

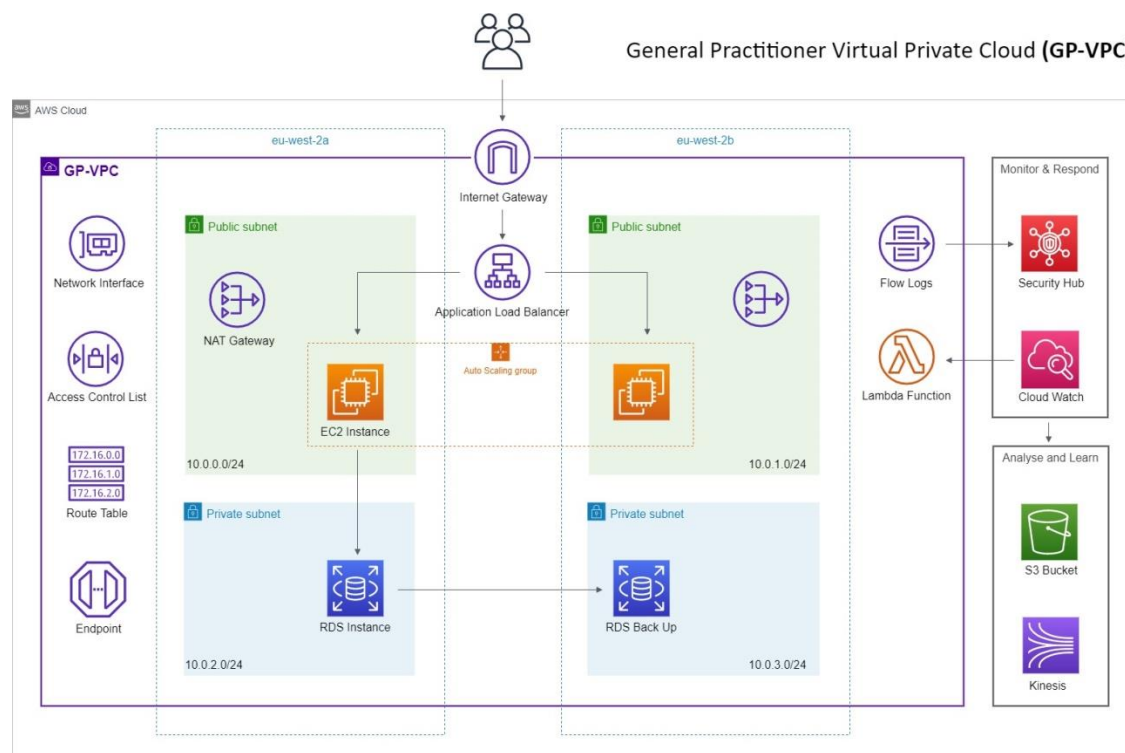
The type of data held by our GP is sensitive and structured with complex relationships. Amazon Relational Database Service (RDS) provides numerous options for a database engine. MySQL, one of the RDS options, can handle millions of requests and has first class security configurations. A detailed description of the configurations is in **Appendix A. section 1.3**. Amazon Elastic Compute (EC2) services include; Auto Scaling and Load Balancing, both are functional requirements, and are discussed in more detail later in this report.

3 Cloud Architecture

Wikipedia (2020) states that, “Cloud Architecture refers to the components and subcomponents required for cloud computing. These components typically consist of a front end platform (fat client, thin client, mobile device), back end platforms (servers, storage), a cloud based delivery, and a network (Internet, Intranet, Intercloud). Combined, these components make up cloud computing architecture”. This section describes the features and components of the GP-VPC Cloud Architecture, and using the CLI, demonstrates a peering connection. The RDS and EC2 are also discussed along with some of the services integrated with the VPC.

3.1 System Walkthrough

The system starts with a user request. The user could be a patient, a medical professional, or the GP himself. Accessing the system requires the user to verify their identity using Identity and Access Management (IAM) and Multi-Factor Authentication (MFA). Once the user is authorised and their request is processed, they are directed to the Internet Gateway.



The Internet Gateway acts as a bridge between the VPC and the Internet. The VPC is protected by Shield, which is a managed service provided for free by AWS. Shield protects against Distributed Denial of Service (DDoS) attacks.

The request must then pass through the Application Load Balancer, which directs incoming traffic and increases availability by sending traffic to healthy instances. The EC2 web server receives the request and responds with either the web page or database results requested. The diagram also illustrates some of the components within our VPC. For example, the Route tables, which control traffic going out of the subnets and Access Control Lists, which control traffic entering the subnets. Subnets are used to separate VPC's and may be public or private. Private subnets do not have direct access to the Internet, while public subnets do have direct access. Its important to note that the Application Load Balancer, EC2 and RDS instances are all protected by security groups within the VPC.

The Monitor and Respond section of the system, on the right of the image, is designed to comply with the AWS well architected framework. The Flow Logs, one of the components in our VPC, feeds the Security Hub data on network activity. The Security Hub is a collection of security services combined into one simple centralised interface. A detailed description and architect drawing of the monitor and respond section is in the next section.

The Security Hub then sends the data to CloudWatch where a response is either required or the data is sent to an S3 bucket. If a response is required CloudWatch Events alert the Lambda Function to take action. The action could be shutting down the entire or part of system. The function could also be programmed to send a notification email. An example of system notifications is available in Appendix A. section 1.4.

The final part of the system is data analysis. Kinesis provides a powerful tool that could be used to analyse and improve system performance, cost or security. The architecture is based on an AWS well architected framework principle, learn from mistakes. The Monitor and analyse section is designed to reduce costs, improve performance and efficiency by monitoring and learning from the system.

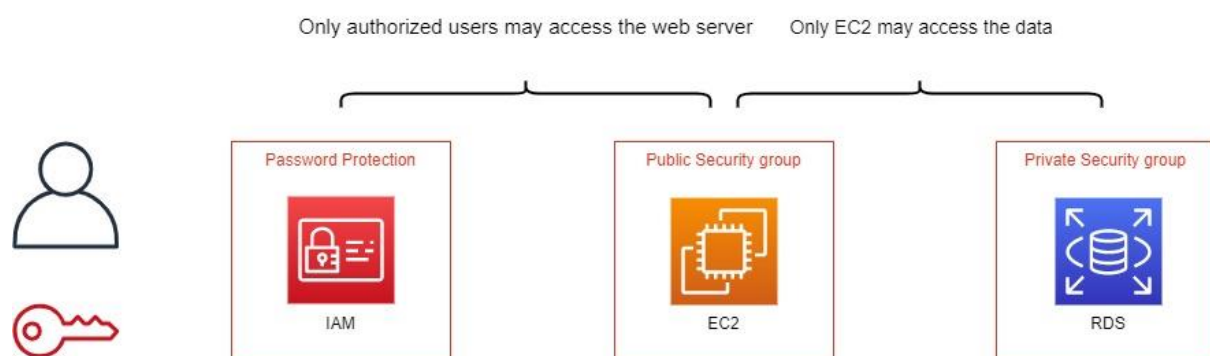
Reliability is hard wired into the system, by using multiple availability zones, an auto scaling group and application load balancer. Even the replicated database ensures fault tolerance. Security is evident on every layer. The user must access the system via IAM and MFA, Shield and WAF protect the resources, while Guard Duty monitors and reports on activity. Indeed, even at the network layer Access Control Lists are used to protect the system.

3.2 Identity and Access Management

At this point, it is important to discuss the difference between authorisation and authentication, IAM manages both. This section will also discuss the shared responsibility model and how it affects the GP-VPC, and its components. UK NHS compliance is also briefly covered.

As previously discussed in the requirements analysis the key characteristic of the data in the GP-VPC is its value. Access to the data is a security priority. IAM and Multi-Factor Authentication (MFA) provide the solution. IAM is a “web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources”. The image below illustrates the role IAM plays in securing the GP data. For each layer of security authorisation is required while MFA manages with authentication. An example of the json files used to control access to data can be found in Appendix E. section 5.2

Example IAM Authorisation



The NHS is an organisation that must adhere to UK compliance regulations and laws. For example, the data protection act of 2018 <https://www.gov.uk/government/collections/data-protection-act-2018>, which enforces strict control of data held by organisations.

3.2.1 The Shared Responsibility Model

The Shared Responsibility Model refers to sharing of security for resources in the cloud. The cloud service provider (CSP) is responsible for the managing of infrastructure or of the cloud and the cloud service customer (CSC) is responsible for securing what is in the cloud, using techniques like least privilege, which is illustrated above in the example IAM authorisation image.

3.3 Virtual Private Cloud (VPC)

The VPC provides the network infrastructure and an additional layer of security for the system. It protects the contents and enables them to communicate with the Internet. There are numerous configurations and micro services that can be utilized in the VPC. The following micro services were configured for use in the GP-VPC.

3.3.1 VPC Components

Micro Service	Function
Application Load Balancer:	traffic is directed to healthy instances
Auto Scaling Group:	adjust capacity to meet demand
NAT Gateway:	enable instances in a private subnet to connect to the internet
Internet Gateway:	allows communication between instances in the VPC and the internet
Subnets:	private and public range of IP addresses
Route Table:	Specify how network traffic is directed inside the VPC
Network Interface:	web server a public IP address
Security Groups:	control inbound and outbound traffic for the instances
Network ACL's	control inbound and specify deny for outbound traffic for subnets
Flow Logs:	Capture the traffic that flows in the VPC
Endpoints	privately connect to a VPC

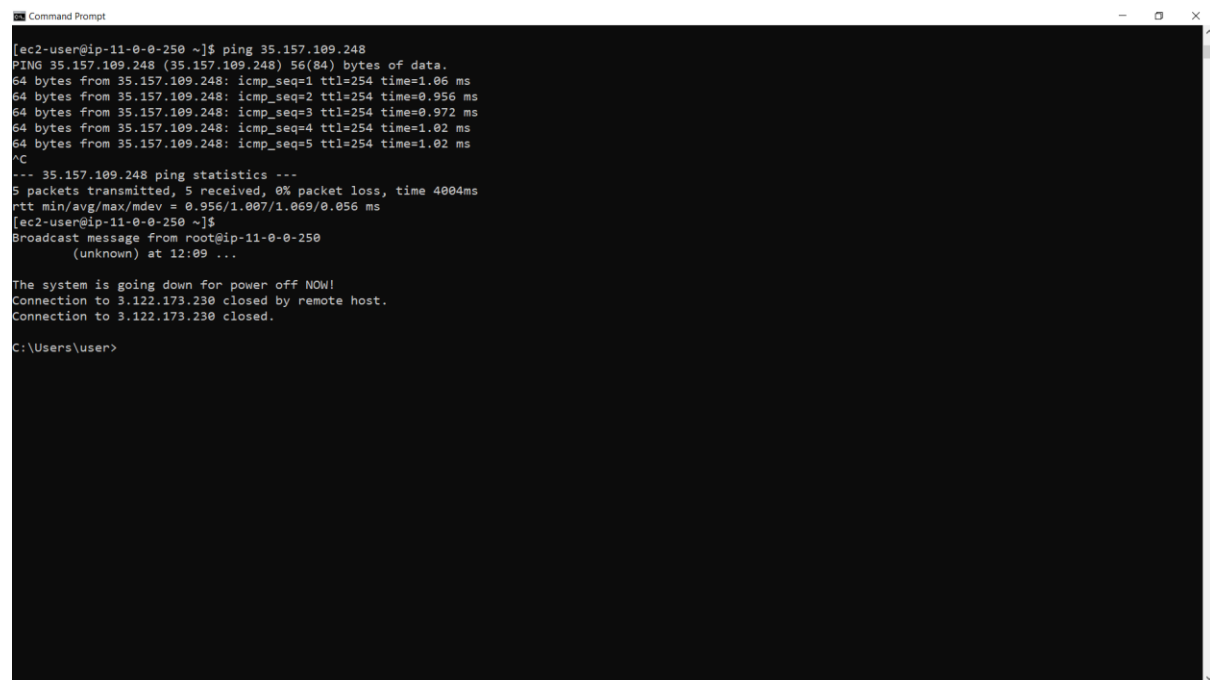
It is important to distinguish between ACL's and Security groups, which both play an important role in securing the subnets and instances within the GP-VPC. Another important component is the NAT Gateway it is responsible for connecting the private subnets with the Internet. It is interesting to note that AWS charge 52p an hour for a NAT Gateway that is not in attached to an instance. Indeed, the instance would not likely cost that much if it were left to run. Flow logs should also be mentioned as they are used, along with other data capture techniques, to aggregate data, and share it with other services such as Security Hub and CloudWatch.

The application load balancer and auto scaling groups are discussed in more detail later in this report. A detailed table listing all the components of the VPC is available in Appendix E.

3.3.2 VPC Peering

In order for the GP to use the VPC to communicate with other GP's or NHS staff a peering connection was created. Peering allows one VPC to communicate with another as if they were instances in the same VPC. This means that if every GP in the country had a VPC, their systems would be able to communicate with each other. The image below demonstrates the VPC Peering Connection, where a ping is sent by one VPC, and received by another. Test documentation in Appendix A, section 1.3

Peering connection



```
Command Prompt
[ec2-user@ip-11-0-0-250 ~]$ ping 35.157.109.248
PING 35.157.109.248 (35.157.109.248) 56(84) bytes of data.
64 bytes from 35.157.109.248: icmp_seq=1 ttl=254 time=1.06 ms
64 bytes from 35.157.109.248: icmp_seq=2 ttl=254 time=0.956 ms
64 bytes from 35.157.109.248: icmp_seq=3 ttl=254 time=0.972 ms
64 bytes from 35.157.109.248: icmp_seq=4 ttl=254 time=1.02 ms
64 bytes from 35.157.109.248: icmp_seq=5 ttl=254 time=1.02 ms
^C
--- 35.157.109.248 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 0.956/1.007/1.069/0.056 ms
[ec2-user@ip-11-0-0-250 ~]$
Broadcast message from root@ip-11-0-0-250:
        (unknown) at 12:09 ...

The system is going down for power off NOW!
Connection to 3.122.173.230 closed by remote host.
Connection to 3.122.173.230 closed.

C:\Users\user>
```

There are numerous types of configurations and scenarios in which VPC Peering can be of business value. The image above illustrates a one to one connection; however a one to multiple connections would be required for the GP to successfully benefit from the service. For example, if the NHS database could communicate with the GP-VPC database then the results of all the tests taken could be instantly aggregated and analysed. The benefits of having test results from each GP in the country delivered to the NHS in real time, for analysis and research are clear and numerous.

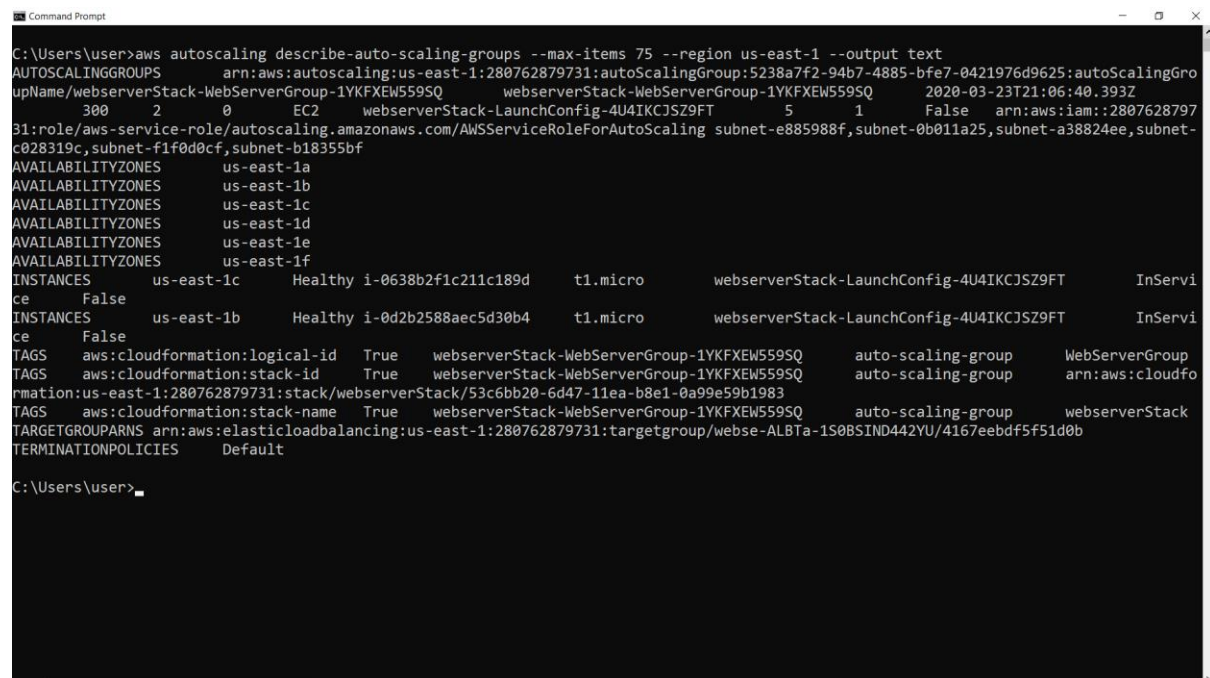
There are also various types of connection available using AWS, which should be investigated for future development. These include;

- Transit Gateway
- Direct Connect
- Virtual Private Networks (VPN)

3.4 Elastic Compute Cloud (EC2)

The EC2 Instance was configured as a web server to interface with the database, the script used in this configuration can be found in Appendix C. section 3.2. The website is a resource, where virus updates and best practices for health can be organised and shared to millions in seconds. A detailed description of the configuration and development is documented in Appendix A. section 1.3. The image below is the output from an aws cli command, listing auto scaling groups.

aws autoscaling describe-auto-scaling-groups



```
C:\Users\user>aws autoscaling describe-auto-scaling-groups --max-items 75 --region us-east-1 --output text
AUTOSCALINGGROUPS    arn:aws:autoscaling:us-east-1:280762879731:autoScalingGroup:5238a7f2-94b7-4885-bfe7-0421976d9625:autoScalingGroup
upName/webserverStack-WebServerGroup-1YKF5EW559SQ    webserverStack-WebServerGroup-1YKF5EW559SQ    2020-03-23T21:06:40.393Z
300    2    0    EC2    webserverStack-LaunchConfig-4U4IKCJSZ9FT    5    1    False    arn:aws:iam::2807628797
31:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling    subnet-e885988f,subnet-0b011a25,subnet-a38824ee,subnet-
c028319c,subnet-f1f0d0cf,subnet-b18355bf
AVAILABILITYZONES    us-east-1a
AVAILABILITYZONES    us-east-1b
AVAILABILITYZONES    us-east-1c
AVAILABILITYZONES    us-east-1d
AVAILABILITYZONES    us-east-1e
AVAILABILITYZONES    us-east-1f
INSTANCES    us-east-1c    Healthy    i-0638b2f1c211c189d    t1.micro    webserverStack-LaunchConfig-4U4IKCJSZ9FT    InServi
ce    False
INSTANCES    us-east-1b    Healthy    i-0d2b2588aec5d30b4    t1.micro    webserverStack-LaunchConfig-4U4IKCJSZ9FT    InServi
ce    False
TAGS    aws:cloudformation:logical-id    True    webserverStack-WebServerGroup-1YKF5EW559SQ    auto-scaling-group    WebServerGroup
TAGS    aws:cloudformation:stack-id    True    webserverStack-WebServerGroup-1YKF5EW559SQ    auto-scaling-group    arn:aws:cloudfo
rmation:us-east-1:280762879731:stack/webserverStack/53c6bb20-6d47-11ea-b8e1-0a90e59b1983
TAGS    aws:cloudformation:stack-name    True    webserverStack-WebServerGroup-1YKF5EW559SQ    auto-scaling-group    webserverStack
TARGETGROUPPARNS    arn:aws:elasticloadbalancing:us-east-1:280762879731:targetgroup/webse-ALBTA-1S0BSIND442YU/4167eebdf5f51d0b
TERMINATIONPOLICIES    Default
C:\Users\user>
```

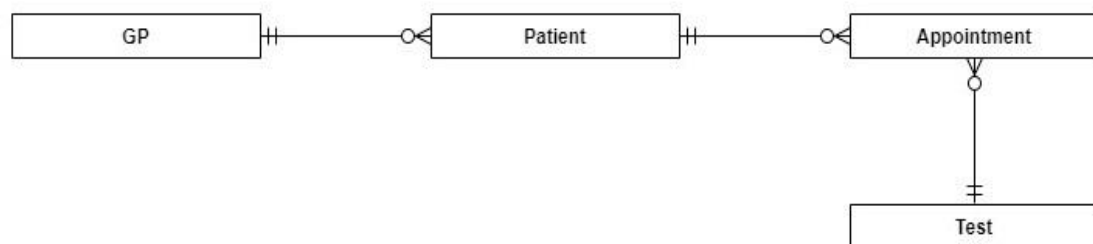
3.5 Relational Database Service (RDS)

EC2 may also be configured to host a relational database. This option was considered and rejected based on performance; RDS is faster, and fault tolerance: RDS includes the option to deploy across multiple availability zones (Multi -AZ). A comprehensive comparison was published by NetApp (2017), which states that RDS, *“provides high availability and failover support for DB instances with Multi-AZ deployments. Multi-AZ deployments for MySQL use Amazon technology, while a hosted Amazon EC2 MySQL database you can use partial replication, Global Transaction Identifier replication, or traditional statement-based replication”*. The publication goes on to discuss the pricing options. The EC2 hosted database is less expensive, however with the additional benefits of fault tolerance and performance, RDS is better value for money.

Although RDS provides its own security, IAM can be configured to provide access to users, adding an additional layer of security and flexibility. All data is encrypted while at rest in the RDS automatically. RDS features also include CloudWatch Logs, which can be generated and used for security analysis, snapshots are taken for back up and disaster recovery and being a fully managed service means less hassle for the GP to ensure patches are up to date and security measures are compliant.

To demonstrate how the RDS, in the GP-VPC could be used, to add business value, a database was designed and based around the need for testing in the country. It is important to test so that the spread of the virus can be tracked. The ERD in Appendix B. section 2.2 illustrates the database design and the image below is an extract from the GP-ERD. A detailed description of the development, deployment and configuration can be found in Appendix A section 1.3.

GP-ERD



The GPDB select statement below is a query written for the database. The query gathers data from all four tables and prints the result. The name of the GP, and patients that tested true for the virus, contact details, age, appointment date and test result. Running the query reveals two patients were found to have the virus. The GP can immediately contact them to arrange treatment.

The above statement provides another example why RDS was chosen to manage the data stored by the GP. A Relational Database Management System (RDMS) is designed to manage complex queries. The full SQL statement for the database can be found in Appendix B.

GPDB Select Statement

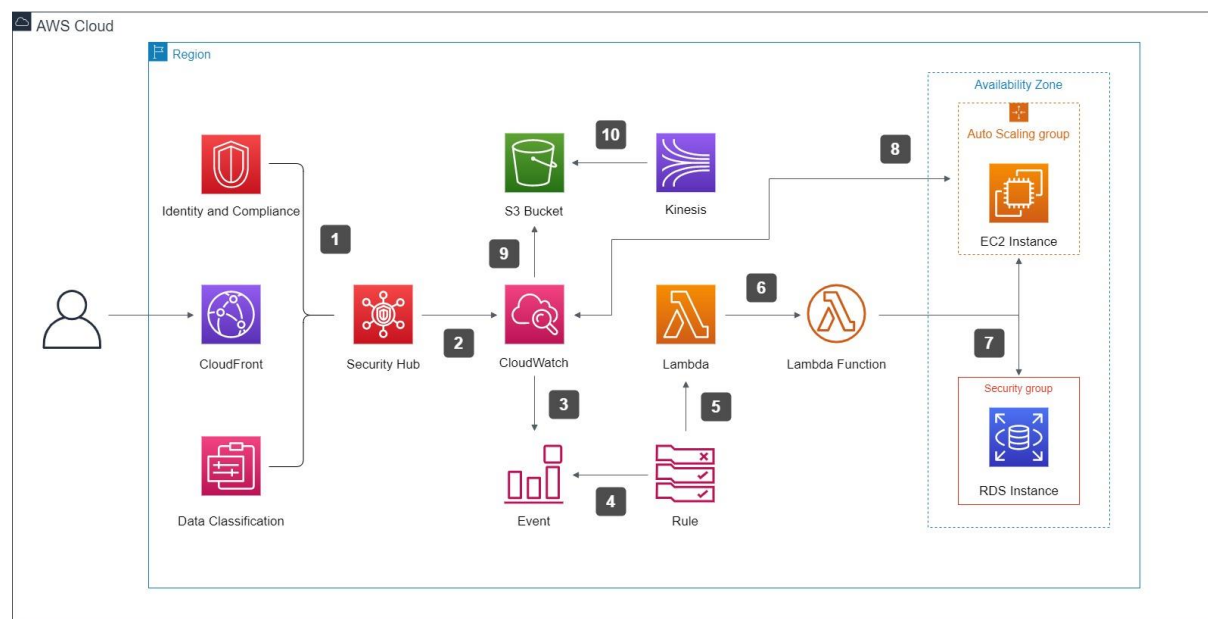
```
SELECT IF (result, 'true', 'false') result, DATE_FORMAT(appointmentDate,'%y-%m-%d') as
'Appointment Date', patient.name as Patient, patient.phone as Contact, YEAR(CURDATE())-
YEAR(dob) as Age, gp.name as GP
FROM test, appointment, patient, gp
WHERE (test.testID=appointment.testID AND patient.patientID=appointment.patientID) AND
(result=true) AND (gp.gpid=patient.gpid);
```

3.6 Monitor and Respond Overview

The diagram below details how the monitor and respond section of our GP-VP, is architected. The diagram also, provides an overview of the entire system and how the VPC integrates. The model was developed with the well architected framework as a guide. For example, automation is an operational excellence objective and monitoring is a performance principle. Indeed security, reliability and cost are all considered in the design and functionality of the system. A full screen view of this architecture is available in Appendix A section 1.2

Cloud Architecture

Overview



Monitor and Respond

1. Identity and Compliance, Data Classification and CloudFront metrics sent to the Security Hub
2. The findings are sent to CloudWatch where
3. An event is triggered
4. A rule (for example, unauthorised access)
5. That triggers Lambda
6. Lambda invokes the function
7. The function takes action to neutralise the threat

Performance and Reliability

8. Health Checks and Auto Scaling

Analyse and Learn

9. CloudWatch sends logs to a bucket
10. Kinesis analyses logs in the bucket

AWS (2016) recommend centralised security as best practice, Security Hub meets that objective. Along with some of the other security services, like Macie and Inspector, IAM Access Analyser is integrated with Security Hub. GuardDuty, which actively monitors the VPC and ensures against DDoS attacks, is also included in the Hub. The management and governance icon refers to the data classification and compliance structures inherited from the industry. The data, classification and compliance are processed in the Security Hub and findings shared with CloudWatch.

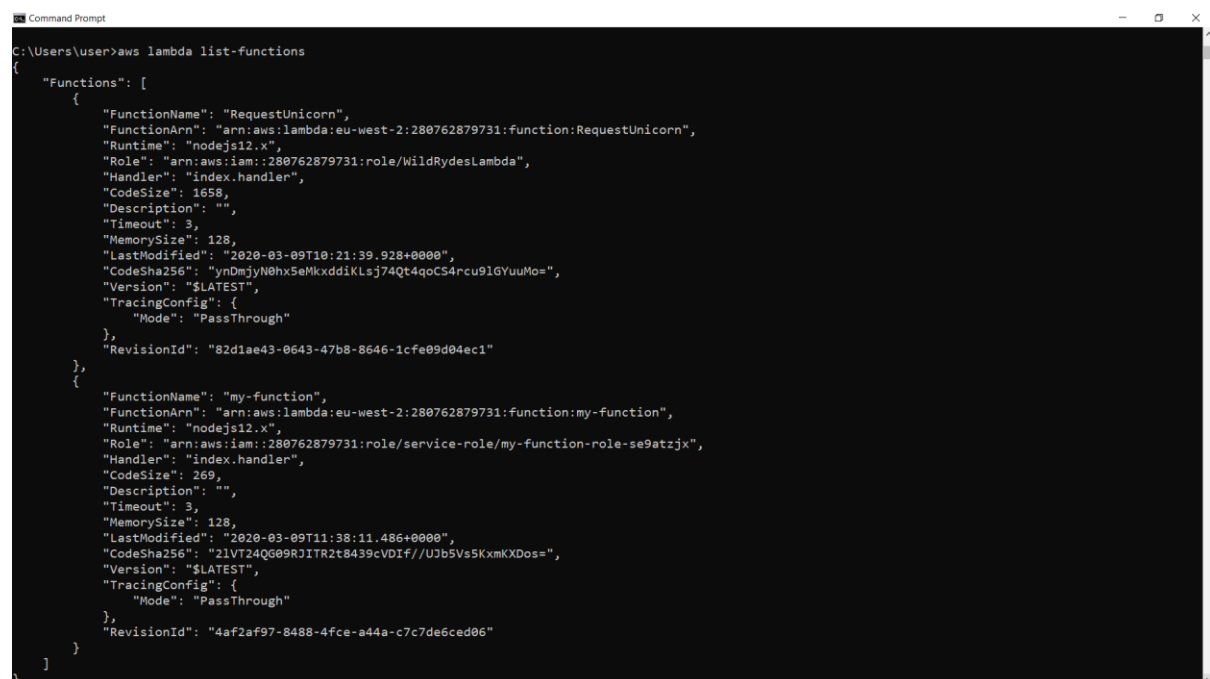
3.6.1 CloudWatch

CloudWatch is a powerful service it is integrated with almost everything AWS and works in conjunction with CloudTrail to ensure all actions a user makes are recorded. CloudWatch processes log files to determine if any rules were broken. If an event is triggered by the data classification rules, then Lambda is invoked and the system responds with an action. Otherwise, the logs are sent to an S3 Bucket and used to analyse and learn. CloudWatch also has a direct relationship with the auto scaling instance. It monitors health and makes adjustments accordingly.

3.6.2 Lambda

Once a security threat has been detected and the Lambda function is invoked the action could be stopping the instance and starting another. Indeed Lambda functionality is able to perform almost any task required from system maintenance to managing the entire system. Evolving with the system and responding to system events are operational excellence objectives.

List of lambda functions via AWS CLI



```
Command Prompt
C:\Users\user>aws lambda list-functions
{
  "Functions": [
    {
      "FunctionName": "RequestUnicorn",
      "FunctionArn": "arn:aws:lambda:eu-west-2:280762879731:function:RequestUnicorn",
      "Runtime": "nodejs12.x",
      "Role": "arn:aws:iam::280762879731:role/WildRydesLambda",
      "Handler": "index.handler",
      "CodeSize": 1658,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2020-03-09T10:21:39.928+0000",
      "CodeSha256": "ynDmjyN0hx5eMkxddiKlsj74Qt4qoCS4rcu91GYuuMo=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "82d1ae43-0643-47b8-8646-1cfe09d04ec1"
    },
    {
      "FunctionName": "my-function",
      "FunctionArn": "arn:aws:lambda:eu-west-2:280762879731:function:my-function",
      "Runtime": "nodejs12.x",
      "Role": "arn:aws:iam::280762879731:role/service-role/my-function-role-se9atzjx",
      "Handler": "index.handler",
      "CodeSize": 269,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2020-03-09T11:38:11.486+0000",
      "CodeSha256": "21VT24QG09RJITRZt8439cVDIf//U3b5Vs5KxmKXDos=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "4af2af97-8488-4fce-a44a-c7c7de6ced06"
    }
  ]
}
```

4 Critical Analysis

The AWS cloud has many advantages however; the cost management interface and pricing structure seem to be deliberately inefficient and complicated. For an entrepreneur to set up and manage a system they will need to spend a significant portion of their time managing costs. Many of the costs are hidden or even misleading.

The GP-VPC system is robust, reliable and fault tolerant, however, the system should be implemented using a server less architecture. This could drastically reduce costs and time managing the system. On the other hand, the setting up the server less system could be more complicated.

5 Conclusion

AWS provide a first class, cost effective and fault tolerant platform to launch, monitor and manage services. Almost everything is integrated, and if its not currently, it most likely will be soon. The pricing and cost management tools are important to understand and utilise, as the value of operating in the cloud is dependent on cost, along with security, availability and fault tolerance.

CloudFormation provides a valuable service that allows developers and administrators to manage deploy and document the services provisioned efficiently and accurately. It manages Elastic Beanstalk and Lightsail applications, making it an extremely powerful tool. With CloudFormation a single line of code can start an entire empire.

CloudWatch and CloudTrail allow customers to track, monitor and notify. The logs generated can be used for security, marketing, or to improve the existing system, by analysing patterns and making design decisions based on learning.

Combining these services with a VPC gives business owners and developers the opportunity to use infrastructure worth billion, for pennies a week. The VPC acts as the foundation for secure, reliable, efficient, compute power or data storage.. Although complicated in technological terms, in real terms the VPC is simple. Cloud Computing technologies such as virtualisation and advancements in Data Center design and structure ensure services like the VPC will only gain in popularity.

Reference List

AWS (2019) Well Architected Framework [online]. Available from:

https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf [14th February 2020]

AWS (2016) re:Invent 2016: IAM Best Practices to live by SAC317 [online]. Available

from: <https://www.youtube.com/watch?v=SGntDzEn30s&feature=youtu.be> [1st March 2020]

AWS (2018) Security Pillar [online]. Available from:

<https://d1.awsstatic.com/whitepapers/architecture/AWS-Security-Pillar.pdf> [27th February 2020]

Rountree, D. and Castrillo, I. (2013). The basics of cloud computing: Understanding the fundamentals of cloud computing in theory and practice

GOV.uk (2018) Data Protection act 2018 [online]. Available from:

<https://www.gov.uk/government/collections/data-protection-act-2018> [7th April 2020]

AWS (2020-c) What Is Amazon Macie? [online]. Available from:

<https://docs.aws.amazon.com/macie/latest/userguide/what-is-macie.html> [22nd March 2020]

NetApp (2017) MySQL Database Migration: EC2-Hosted vs. RDS [online]. Available from:

<https://cloud.netapp.com/blog/migrating-mysql-database-ec2-hosted-amazon-rds> [7th April 2020]

Wikipedia (2020) Cloud computing architecture [online]. Available from:

https://en.wikipedia.org/wiki/Cloud_computing_architecture [8th April 2020]

AWS (2020b) Automating AWS Security Hub with CloudWatch Events [Online]. Available from:

<https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-cloudwatch-events.html>
[22nd March 2020]

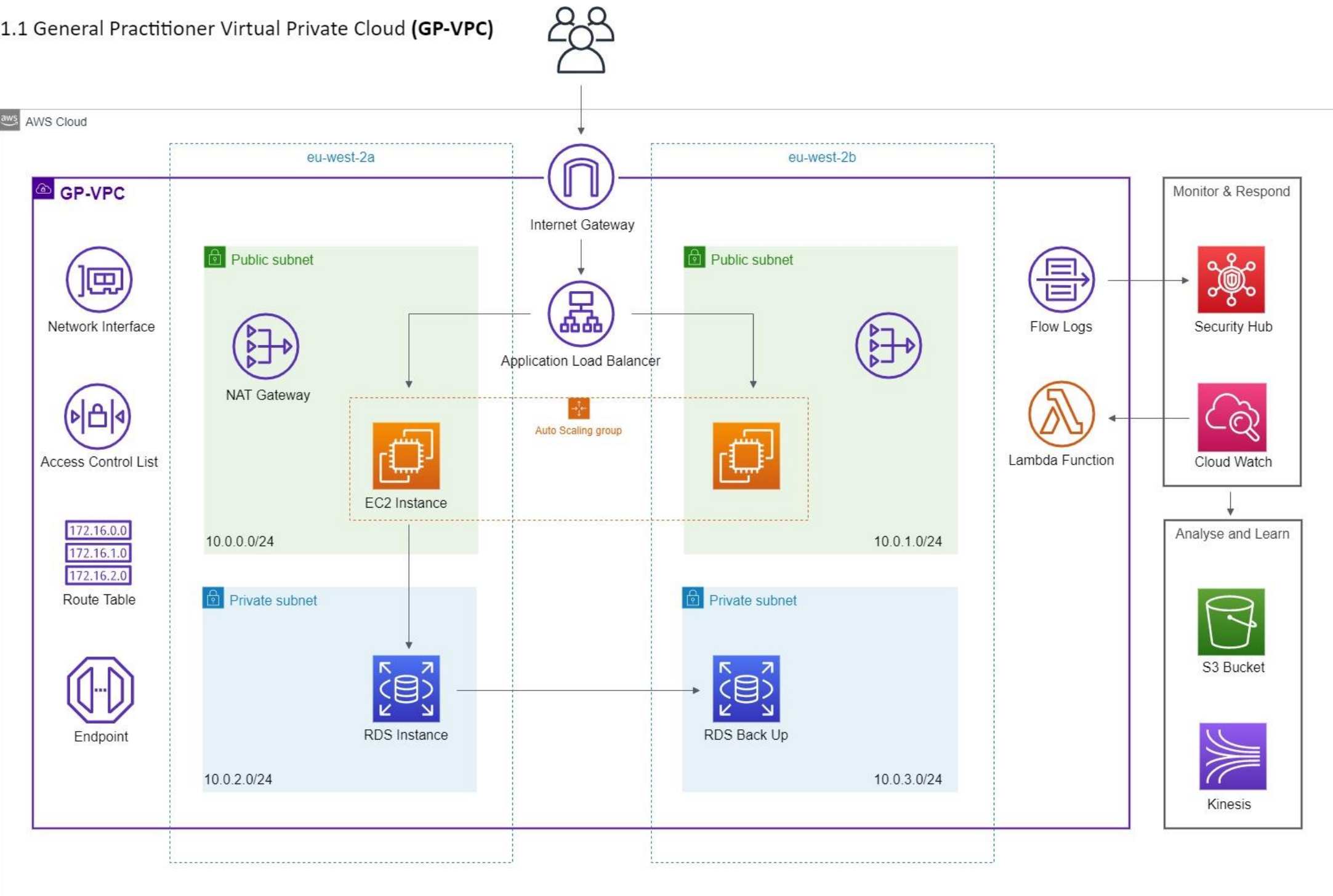
AWS (2019b) AWS Web Hosting Best Practices [online]. Available from:

<https://d1.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf> [16th March 2020]

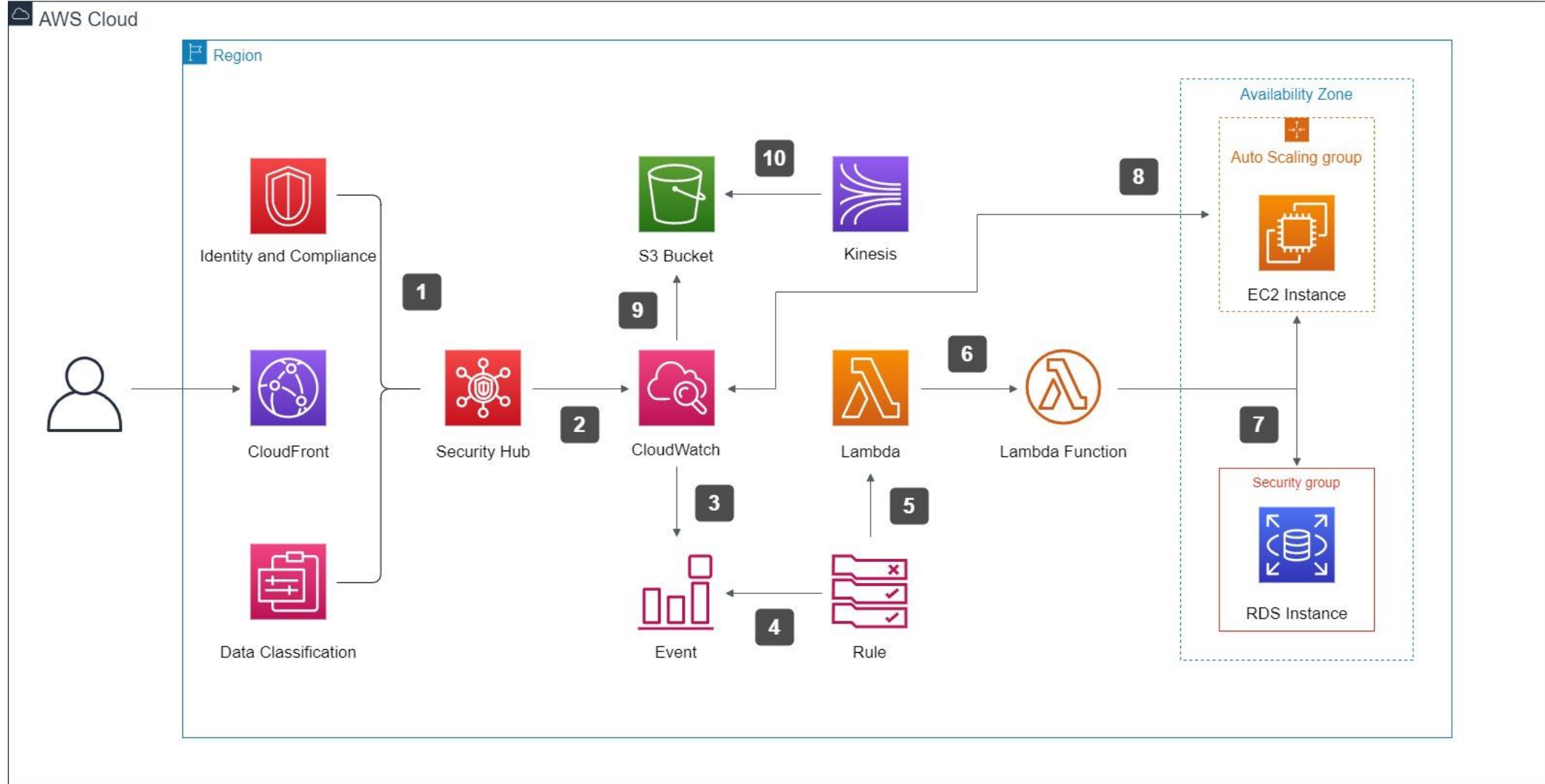
Cloud Development

Appendix A

1.1 General Practitioner Virtual Private Cloud (GP-VPC)



1.2 Monitor and Respond - System Overview



1.3 Console Development

As recommended by the AWS well architected framework, at least two subnets are required, one to be public facing and one private. The public facing subnet will be used as a web server, and the private subnet used to add an additional layer of security for the secure database.

1. Create a VPC configuration - vpc-0c4cbe0355a4775e6

The screenshot shows the 'Step 2: VPC with Public and Private Subnets' configuration page in the AWS Management Console. The page is for creating a VPC named 'gp-vpc'. Key configurations include:

- IPv4 CIDR block:** 10.0.0.0/16 (65531 IP addresses available)
- IPv6 CIDR block:** No IPv6 CIDR Block (selected)
- Public subnet's IPv4 CIDR:** 10.0.0.0/24 (251 IP addresses available)
- Public subnet name:** GP Public Subnet
- Private subnet's IPv4 CIDR:** 10.0.1.0/24 (251 IP addresses available)
- Private subnet name:** GP Private Subnet 1
- Instance type:** t2.small
- Key pair name:** No key pair
- Enable DNS hostnames:** Yes
- Hardware tenancy:** Default

Buttons at the bottom right: Cancel and Exit, Back, Create VPC.

2. Create additional Subnet – subnet-03dbc15e016449caa

The screenshot shows the 'Create subnet' page in the AWS Management Console. The page is for creating a new subnet. Key configurations include:

- Name tag:** GP Private 2
- VPC:** vpc-0c4cbe0355a4775e6
- Availability Zone:** eu-west-2b
- IPv4 CIDR block:** 10.0.2.0/24

A table shows the VPC CIDRs and their status:

VPC CIDRs	CIDR	Status	Status Reason
	10.0.0.0/16	associated	

Buttons at the bottom right: Cancel, Create.

3. Create Security Groups for public and private subnets - sg-0c195f869438b803c

Security Groups > Create security group

Create security group

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group fill in the fields below.

Security group name*

Description*

VPC

* Required

Cancel Create

VPC ID	Name tag	Owner
vpc-0c4cbe0355a4775e6	gp-vpc	280762879731

The security groups are required by default. However, they do add an additional layer of security. For example, the private security group can be configured to only allow inbound traffic from the EC2 instance. This means that any would be hacker that access the EC2 instance would still not gain access to the data.

4. Create Inbound rules for SSH access

Security Groups > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	My IP 188.31.92.221/32	SSH for admin/development
HTTP	TCP	80	Anywhere 0.0.0.0/0, ::0	Public access

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

* Required

Cancel Save rules

5. Create Private subnet

The screenshot shows the AWS VPC console. On the left is a navigation menu with categories like Virtual Private Cloud, Security, and Virtual Private Network. The main content area is titled 'Create security group'. It shows a table of existing security groups:

Name	Group ID	Group Name	VPC ID	Type	Description	Owner
gp-security	sg-04d77cf2dbd55a8d4	gp-db-security-group	vpc-0c4cbe0355a...	EC2-VPC	rds security group	280762879731
gp-security	sg-0c195f869438b...	gp-security-group	vpc-0c4cbe0355a...	EC2-VPC	public facing subn...	280762879731

Below the table, the configuration for the selected security group 'sg-04d77cf2dbd55a8d4' is shown. The 'Inbound Rules' tab is active, displaying a single rule:

Type	Protocol	Port Range	Source	Description
MYSQL/Aurora	TCP	3306	sg-0c195f869438b803c	connect web server and db

Create the Database

RDS endpoint: `gp-db.cpyoaulck67v.eu-west-2.rds.amazonaws.com`

The endpoints are used to connect to the database. The endpoint could be public or private. In this case the endpoint is private. The data stored by the GP must be protected at every layer.

6. Create DB Subnet

The screenshot shows the AWS RDS console 'Create DB subnet group' page. It includes a sidebar for Amazon RDS navigation. The main content area has the following sections:

- Subnet group details**: Fields for Name (gp-db-subnet-group), Description (GP Database Subnet Group), and VPC (gp-vpc (vpc-0c4cbe0355a4775e6)).
- Add subnets**: A button to 'Add all the subnets related to this VPC' and a dropdown for 'Availability zone' (eu-west-2a).

7. Complete the configuration

The configurations set on this page are critical to the performance and reliability of the database. Indeed the costs and security pillars should not be overlooked. It is important to understand each configuration option.

On this page users can configure IAM groups that have access to the data, decide where and how to store log files and ensure the database is deployed across multiple availability zones,

- a. Subnet group - gp-db-subnet-group
- b. Add VPC Security group
- c. Add Subnets
- d. Publicly accessible – NO
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.
- e. Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC.
- f. Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.
Choose
- g. IAM role
- h. Deletion protection
- i. Maintenance
- j. Backup
- k. Monitoring
- l. Log exports – to CloudWatch

RDS Configuration

The screenshot shows the 'Connectivity' configuration page in the AWS Management Console. The page is titled 'Connectivity' and includes a 'Virtual Private Cloud (VPC)' section with a dropdown menu showing 'gp-vpc (vpc-0c4be0355a4775e6)'. Below this is a warning box stating: 'After a database is created, you can't change the VPC selection.' The 'Additional connectivity configuration' section includes a 'Subnet group' dropdown showing 'gp-db-subnet-group'. The 'Publicly accessible' section has two radio buttons: 'Yes' and 'No'. The 'No' option is selected, with a note: 'RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.' The 'VPC security group' section has two radio buttons: 'Choose existing' and 'Create new'. The 'Choose existing' option is selected, and a dropdown menu shows 'gp-db-security-group'.

aws Services Resource Groups

Richard @ uel London Support

Connectivity

Virtual Private Cloud (VPC) [Info](#)
VPC that defines the virtual networking environment for this DB instance.
gp-vpc (vpc-0c4be0355a4775e6)
Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change the VPC selection.

Additional connectivity configuration

Subnet group [Info](#)
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.
gp-db-subnet-group

Publicly accessible [Info](#)

☐ Yes
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

☒ No
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)

☒ Choose existing
Choose existing VPC security groups

☐ Create new
Create new VPC security group

Existing VPC security groups
Choose VPC security groups
gp-db-security-group

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

8. Complete the settings

The screenshot shows the 'Settings' page for an Amazon RDS database instance. The page is divided into several sections:

- DB instance identifier**: A text input field containing 'gp-db'. Below it, a note states: 'The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.'
- Credentials Settings**:
 - Master username**: A text input field containing 'gp-db-master'. Below it, a note states: 'Type a login ID for the master user of your DB instance. 1 to 16 alphanumeric characters. First character must be a letter.'
 - Auto generate a password**: A checkbox that is currently unchecked. Below it, a note states: 'Amazon RDS can generate a password for you, or you can specify your own password.'
 - Master password**: A text input field containing '*****'. Below it, a note states: 'Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), " (double quote) and @ (at sign).'
 - Confirm password**: A text input field containing '*****'.
- DB instance size**:
 - DB instance class**: A text input field containing 'db.t2.micro'. Below it, a note states: 'Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.'
 - Standard classes (includes m classes)**: A link to view more options.

The footer of the console shows 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

Once the database is created, the EC2 instance that will be configured to serve the content can be provisioned. There are numerous options to do this, however the next section details the stages using the management console

9. The database is created

The screenshot shows the 'Amazon RDS' page in the AWS Management Console. The left sidebar contains a navigation menu with options like 'Dashboard', 'Databases', 'Performance Insights', 'Snapshots', 'Automated backups', 'Reserved instances', 'Subnet groups', 'Parameter groups', 'Option groups', 'Events', 'Event subscriptions', 'Recommendations', and 'Certificate update'. The main content area displays the 'Databases' list. At the top, there are two status messages: 'Creating database gp-db. Your database might take a few minutes to launch.' and 'Deleting DB instance rds-mysql-10mintutorial.'.

The 'Databases' table has the following columns: DB identifier, Role, Engine, Region & AZ, Size, Status, and CPU. The table contains three entries:

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU
gp-db	Instance	MySQL Community	eu-west-2c	db.t2.micro	Creating	-
gpdb	Instance	MySQL Community	eu-west-2a	db.t2.micro	Available	1.3
rds-mysql-10mintutorial	Instance	MySQL Community	eu-west-2a	db.t2.micro	Deleting	1.3

The footer of the console shows 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

Appendix A section 1.4 documents the process of creating an instance using CloudFormation.

Create EC2 instance and install web server

Instance id: - *i-00c6ca195e3f92dd3*

1. Chose AMI

The screenshot shows the AWS Management Console interface for 'Step 1: Choose an Amazon Machine Image (AMI)'. The breadcrumb trail at the top indicates the steps: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. The page title is 'Step 1: Choose an Amazon Machine Image (AMI)' with a 'Cancel and Exit' link. Below the title, a search bar prompts the user to 'Search for an AMI by entering a search term e.g. "Windows"'. A 'Quick Start' sidebar on the left lists 'My AMIs', 'AWS Marketplace', and 'Community AMIs', with a 'Free tier only' filter. The main content area displays a list of AMIs. The first four are: 'Amazon Linux 2 AMI (HVM), SSD Volume Type' (ami-0389b2a3c4948b1a0), 'Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type' (ami-050b8344d7708114b), 'Red Hat Enterprise Linux 8 (HVM), SSD Volume Type' (ami-0a0cb6c7bcb2e4c51), and 'SUSE Linux Enterprise Server 15 SP1 (HVM), SSD Volume Type' (ami-05289a1dd768f016). Each entry includes a description, root device type, virtualization type, and ENA status, along with a 'Select' button. The footer contains a 'Feedback' link, 'English (US)' language selector, and copyright information for 2008-2020 Amazon Web Services, Inc.

The configurations on this page are critical to the performance of the system. This example uses a micro instance to demonstrate the process.

2. Configure the instance

The screenshot shows the AWS Management Console interface for 'Step 3: Configure Instance Details'. The breadcrumb trail at the top indicates the steps: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. The page title is 'Step 3: Configure Instance Details' with a 'Cancel and Exit' link. Below the title, a sub-header 'Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.' is followed by a 'Launch into Auto Scaling Group' link. The configuration form includes several sections: 'Number of instances' (set to 1), 'Purchasing option' (with a checkbox for 'Request Spot instances'), 'Network' (with dropdowns for 'vpc-0c4cbe0355a4775e6 | gp-vpc' and 'subnet-04600ec66b8e1e855 | GP Public Subnet | eu', and buttons for 'Create new VPC' and 'Create new subnet'), 'Auto-assign Public IP' (set to 'Enable'), 'Placement group' (with a checkbox for 'Add instance to placement group'), 'Capacity Reservation' (set to 'Open', with a button for 'Create new Capacity Reservation'), 'IAM role' (set to 'None', with a button for 'Create new IAM role'), 'Shutdown behavior' (set to 'Stop'), 'Enable termination protection' (checkbox), 'Monitoring' (checkbox for 'Enable CloudWatch detailed monitoring' with a note 'Additional charges apply.'), 'Tenancy' (set to 'Shared - Run a shared hardware instance' with a note 'Additional charges will apply for dedicated tenancy.'), and 'T2/T3 Unlimited' (checkbox). At the bottom right, there are 'Cancel', 'Previous', 'Review and Launch', and 'Next: Add Storage' buttons. The footer contains a 'Feedback' link, 'English (US)' language selector, and copyright information for 2008-2020 Amazon Web Services, Inc.

3. Configure Security groups

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☐ Create a new security group
☒ Select an existing security group

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-03900c205327afb7	default	default VPC security group	Copy to new
<input type="checkbox"/> sg-04d77cf2dbd55a6d4	gp-db-security-group	rds security group	Copy to new
<input checked="" type="checkbox"/> sg-0c195f869438b803c	gp-security-group	public facing subnet - security group	Copy to new

Inbound rules for sg-0c195f869438b803c (Selected security groups: sg-0c195f869438b803c)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Public access
HTTP	TCP	80	:::0	Public access
SSH	TCP	22	188.31.92.221/32	SSH for admin/deve...

[Cancel](#) [Previous](#) [Review and Launch](#)

These security groups will be used to access the database. The only access to the database will be via the security groups configured on this page.

For the purpose of this test an additional inbound rule was added allowing the developer to access the database using a private connection

4. After completing the configuration the instance is created

Instances

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IP
gp-security	i-00c6ca195e3f92dd3	t2.micro	eu-west-2c	running	2/2 checks ...	None	ec2-3-10-174-236.eu-w...	3.10.174.236	-

Instance: i-00c6ca195e3f92dd3 (gp-security) Public DNS: ec2-3-10-174-236.eu-west-2.compute.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID: i-00c6ca195e3f92dd3	Public DNS (IPv4): ec2-3-10-174-236.eu-west-2.compute.amazonaws.com	IPv4 Public IP: 3.10.174.236	IPv6 IPs: -
Instance state: running	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Instance type: t2.micro	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Opt-in to AWS Compute Optimizer for recommendations. Learn more	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
VPC ID: vpc-04cbe0355a4775e6 (gp-vpc)	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Subnet ID: subnet-04600ec66b8e1e855 (GP Public Subnet)	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Network interfaces: eth0	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Source/dest. check: True	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Availability zone: eu-west-2c	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Security groups: gp-security-group. view inbound rules. view outbound rules	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Scheduled events: No scheduled events	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
AMI ID: amzn-ami-hvm-2018.03.0.20200206.0-x86_64-gp2 (ami-050b8344d7708114b)	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Platform: -	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
IAM role: -	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -
Key pair name: gp-key-pair	Private DNS: ip-10-0-0-96.eu-west-2.compute.internal	Private IPs: 10.0.0.96	Secondary private IPs: -

5. Connect to the instance

Connection string used to connect to the EC2 instance:

- `ssh -i gp-key-pair.pem ec2-user@ec2-3-10-174-236.eu-west-2.compute.amazonaws.com`

```
ec2-user@ip-10-0-0-96~
C:\Users\user>ssh -i gp-key-pair.pem ec2-user@ec2-3-10-174-236.eu-west-2.compute.amazonaws.com
The authenticity of host 'ec2-3-10-174-236.eu-west-2.compute.amazonaws.com (3.10.174.236)' can't be established.
ECDSA key fingerprint is SHA256:0iX4NPxN0w5Vlf4yLuCndHxSjWtkH7gs5qVr0kNzgy.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-3-10-174-236.eu-west-2.compute.amazonaws.com,3.10.174.236' (ECDSA) to the list of known hosts.

  _ | _ | _ |
  _ | ( _ | _ |
  _ | \ _ | _ |
  _ |  \ _ | _ |

Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
4 package(s) needed for security, out of 10 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-10-0-0-96 ~]$ scp -i gp-key-pair.pem /gp-cloud9/samplepage.php ec2-user@ec2-3-10-174-236.eu-west-2.compute.amazonaws.com
cp: cannot stat '/gp-cloud9/samplepage.php': No such file or directory
[ec2-user@ip-10-0-0-96 ~]$
```

6. Update the server and Install the web server

Update and install syntax

- `sudo yum update -y`
- `sudo yum install -y httpd24 php56 php56-mysql`

```
ec2-user@ip-10-0-0-96~
(6/10): php56-process-5.6.40-1.143.amzn1.x86_64.rpm | 100 kB 00:00:00
(7/10): php56-common-5.6.40-1.143.amzn1.x86_64.rpm | 1.4 MB 00:00:00
(8/10): php56-xml-5.6.40-1.143.amzn1.x86_64.rpm | 345 kB 00:00:00
(9/10): php56-5.6.40-1.143.amzn1.x86_64.rpm | 3.0 MB 00:00:00
(10/10): php56-cli-5.6.40-1.143.amzn1.x86_64.rpm | 4.2 MB 00:00:00
-----
Total | 15 MB/s | 11 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : php56-jsonc-1.3.10-1.20.amzn1.x86_64 1/10
Installing : php56-process-5.6.40-1.143.amzn1.x86_64 2/10
Installing : php56-cli-5.6.40-1.143.amzn1.x86_64 3/10
Installing : php56-xml-5.6.40-1.143.amzn1.x86_64 4/10
Installing : php56-common-5.6.40-1.143.amzn1.x86_64 5/10
Installing : apr-1.5.2-5.13.amzn1.x86_64 6/10
Installing : apr-util-1.5.4-6.18.amzn1.x86_64 7/10
Installing : httpd24-tools-2.4.41-1.88.amzn1.x86_64 8/10
Installing : httpd24-2.4.41-1.88.amzn1.x86_64 9/10
Installing : php56-5.6.40-1.143.amzn1.x86_64 10/10
Verifying : php56-common-5.6.40-1.143.amzn1.x86_64 1/10
Verifying : php56-5.6.40-1.143.amzn1.x86_64 2/10
Verifying : php56-jsonc-1.3.10-1.20.amzn1.x86_64 3/10
Verifying : httpd24-2.4.41-1.88.amzn1.x86_64 4/10
Verifying : php56-process-5.6.40-1.143.amzn1.x86_64 5/10
Verifying : apr-1.5.2-5.13.amzn1.x86_64 6/10
Verifying : php56-cli-5.6.40-1.143.amzn1.x86_64 7/10
Verifying : php56-xml-5.6.40-1.143.amzn1.x86_64 8/10
Verifying : httpd24-tools-2.4.41-1.88.amzn1.x86_64 9/10
Verifying : apr-util-1.5.4-6.18.amzn1.x86_64 10/10

Installed:
httpd24.x86_64 0:2.4.41-1.88.amzn1 php56.x86_64 0:5.6.40-1.143.amzn1

Dependency Installed:
apr.x86_64 0:1.5.2-5.13.amzn1 apr-util.x86_64 0:1.5.4-6.18.amzn1 httpd24-tools.x86_64 0:2.4.41-1.88.amzn1
php56-cli.x86_64 0:5.6.40-1.143.amzn1 php56-common.x86_64 0:5.6.40-1.143.amzn1 php56-jsonc.x86_64 0:1.3.10-1.20.amzn1
php56-process.x86_64 0:5.6.40-1.143.amzn1 php56-xml.x86_64 0:5.6.40-1.143.amzn1

Complete!
[ec2-user@ip-10-0-0-96 ~]$
```

7. Start the server:

Start service syntax

- `sudo service httpd start`

```
ec2-user@ip-10-0-0-96:/var/www/html$ sudo service httpd start
Dependency Installed:
  apr.x86_64 0:1.5.2-5.13.amzn1          apr-util.x86_64 0:1.5.4-6.18.amzn1      httpd24-tools.x86_64 0:2.4.41-1.88.amzn1
  php56-cli.x86_64 0:5.6.40-1.143.amzn1  php56-common.x86_64 0:5.6.40-1.143.amzn1  php56-jsonc.x86_64 0:1.3.10-1.20.amzn1
  php56-process.x86_64 0:5.6.40-1.143.amzn1  php56-xml.x86_64 0:5.6.40-1.143.amzn1

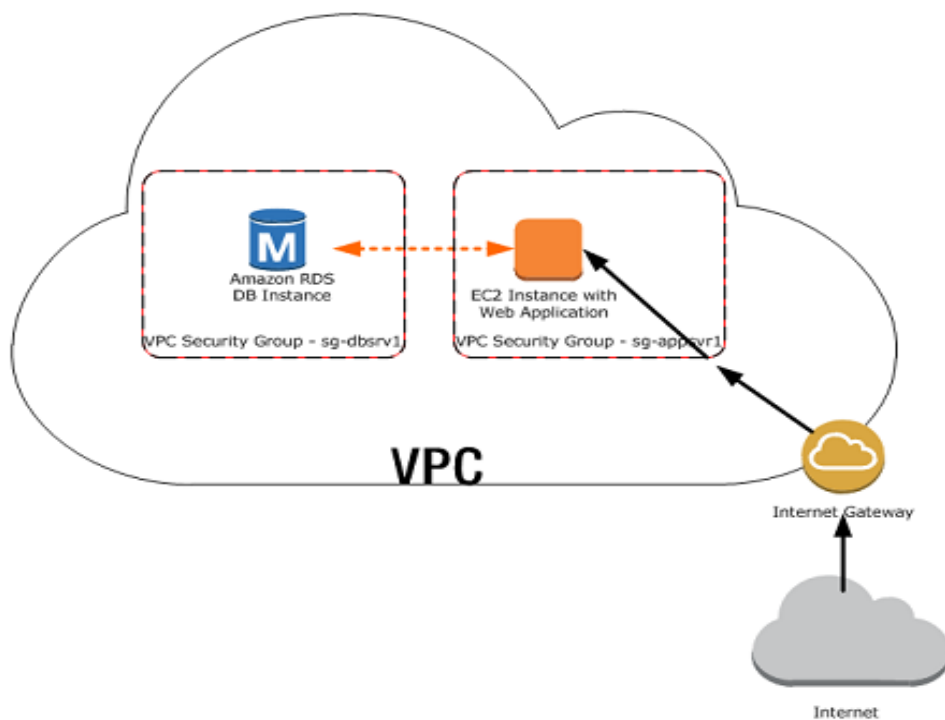
Complete!
[ec2-user@ip-10-0-0-96 ~]$ sudo service httpd start
Starting httpd: [ OK ]
[ec2-user@ip-10-0-0-96 ~]$ sudo chkconfig httpd on
[ec2-user@ip-10-0-0-96 ~]$ sudo groupadd www
[ec2-user@ip-10-0-0-96 ~]$ dir
[ec2-user@ip-10-0-0-96 ~]$ sudo usermod -a -G www ec2-user
[ec2-user@ip-10-0-0-96 ~]$ exit
logout
Connection to ec2-3-10-174-236.eu-west-2.compute.amazonaws.com closed.

C:\Users\User>ssh -i gp-key-pair.pem ec2-user@ec2-3-10-174-236.eu-west-2.compute.amazonaws.com
Last login: Sat Mar 7 15:36:05 2020 from 188.31.92.221.threembb.co.uk

 _ _ | _ _ |
 _ _ | _ _ | Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-10-0-0-96 ~]$ groups
ec2-user wheel www
[ec2-user@ip-10-0-0-96 ~]$ sudo chgrp -R www /var/www
[ec2-user@ip-10-0-0-96 ~]$ sudo chmod 2775 /var/www
[ec2-user@ip-10-0-0-96 ~]$ find /var/www -type d -exec sudo chmod 2775 {} +
[ec2-user@ip-10-0-0-96 ~]$ find /var/www -type f -exec sudo chmod 0664 {} +
[ec2-user@ip-10-0-0-96 ~]$ cd /var/www
[ec2-user@ip-10-0-0-96 www]$ mkdir inc
[ec2-user@ip-10-0-0-96 www]$ cd inc
[ec2-user@ip-10-0-0-96 inc]$ >dbinfo.inc
[ec2-user@ip-10-0-0-96 inc]$ nano dbinfo.inc
[ec2-user@ip-10-0-0-96 inc]$ cd /var/www/html
[ec2-user@ip-10-0-0-96 html]$ >test.php
[ec2-user@ip-10-0-0-96 html]$ nano test.php
[ec2-user@ip-10-0-0-96 html]$ >samplepage.php
[ec2-user@ip-10-0-0-96 html]$ nano samplepage.php
[ec2-user@ip-10-0-0-96 html]$ [ec2-user@ip-10-0-0-96 html]$
```

Virtual Private Cloud (VPC), Relational Database Service (RDS) and Elastic Compute (EC2)



https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/TUT_WebAppWithRDS.html

Peering

- Create 2 VPCs with different cidr blocks and verify internet gateways are present in both
- Create VPC peering connection, accept the request
- Edit route tables: add new route to the public subnets to the VPC connection **pcx** for VPCs

The screenshot shows the AWS Management Console interface for creating and managing route tables. The 'Routes' tab is active, showing a table of routes for the selected route table (rtb-0d8cfa0407105f530). The routes table has columns for Destination, Target, Status, and Propagated. The routes are as follows:

Destination	Target	Status	Propagated
11.0.0.0/16	local	active	No
0.0.0.0/0	lgw-0d0d047cee9dd4200	active	No
10.0.0.0/16	pcx-0e3dd10c9d9240b0f	active	No

Create instances in both VPCs then edit the VPC security group to accept all ICMP

- `aws ec2 associate-address --instance-id i-0621ac048f4 --allocation-id eipalloc-04d999`
- `ssh -i peeringKeyPair.pem ec2-user@3.122.173.230`

Ping the server

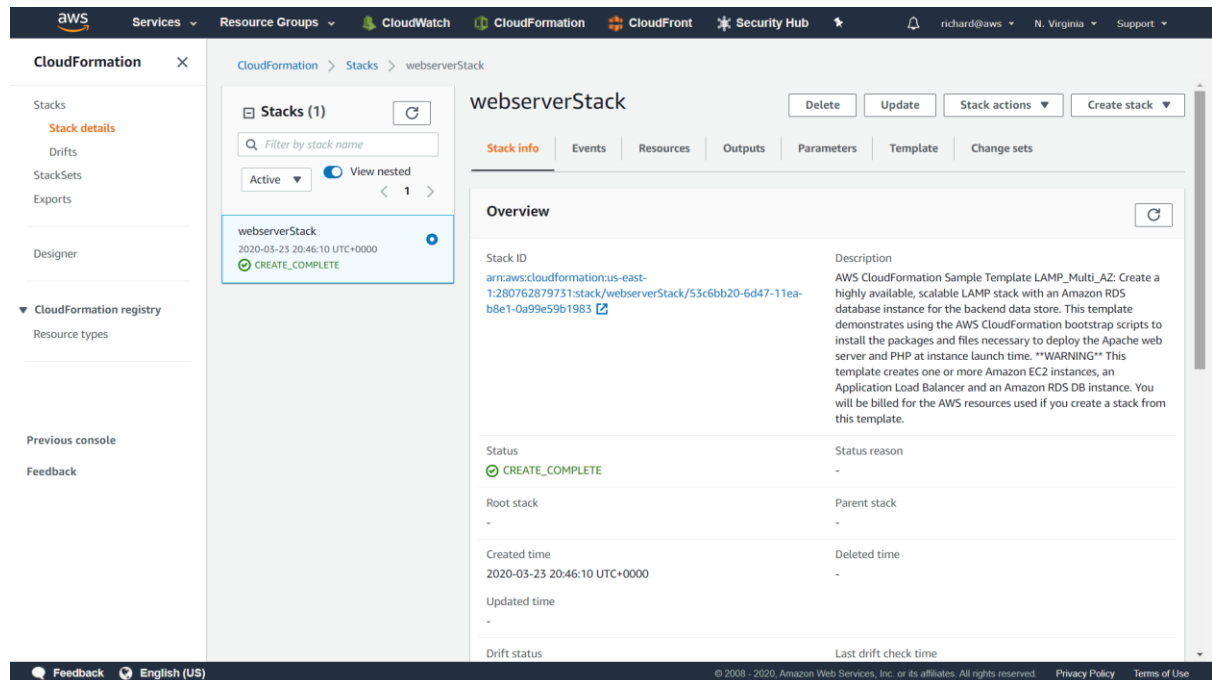
```
Command Prompt
[ec2-user@ip-11-0-0-250 ~]$ ping 35.157.109.248
PING 35.157.109.248 (35.157.109.248) 56(84) bytes of data:
64 bytes from 35.157.109.248: icmp_seq=1 ttl=254 time=1.06 ms
64 bytes from 35.157.109.248: icmp_seq=2 ttl=254 time=0.956 ms
64 bytes from 35.157.109.248: icmp_seq=3 ttl=254 time=0.972 ms
64 bytes from 35.157.109.248: icmp_seq=4 ttl=254 time=1.02 ms
64 bytes from 35.157.109.248: icmp_seq=5 ttl=254 time=1.02 ms
^C
--- 35.157.109.248 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 0.956/1.007/1.069/0.056 ms
[ec2-user@ip-11-0-0-250 ~]$
Broadcast message from root@ip-11-0-0-250:
    (unknown) at 12:09 ...

The system is going down for power off NOW!
Connection to 3.122.173.230 closed by remote host.
Connection to 3.122.173.230 closed.

C:\Users\user>
```


1.4 CloudFormation

Once the basic structure and contents of the system had been decided, CloudFormation was used to deploy the prototype application, and assist in the documentation of the project. . By using CloudFormation to complete the development users are able to see where improvements can be made and test them before implementing them in the live system.



During the Cloud Development Process, two features of the system were not included, the Application Load balancer and the Auto Scaling Group. Both were included in the final template.

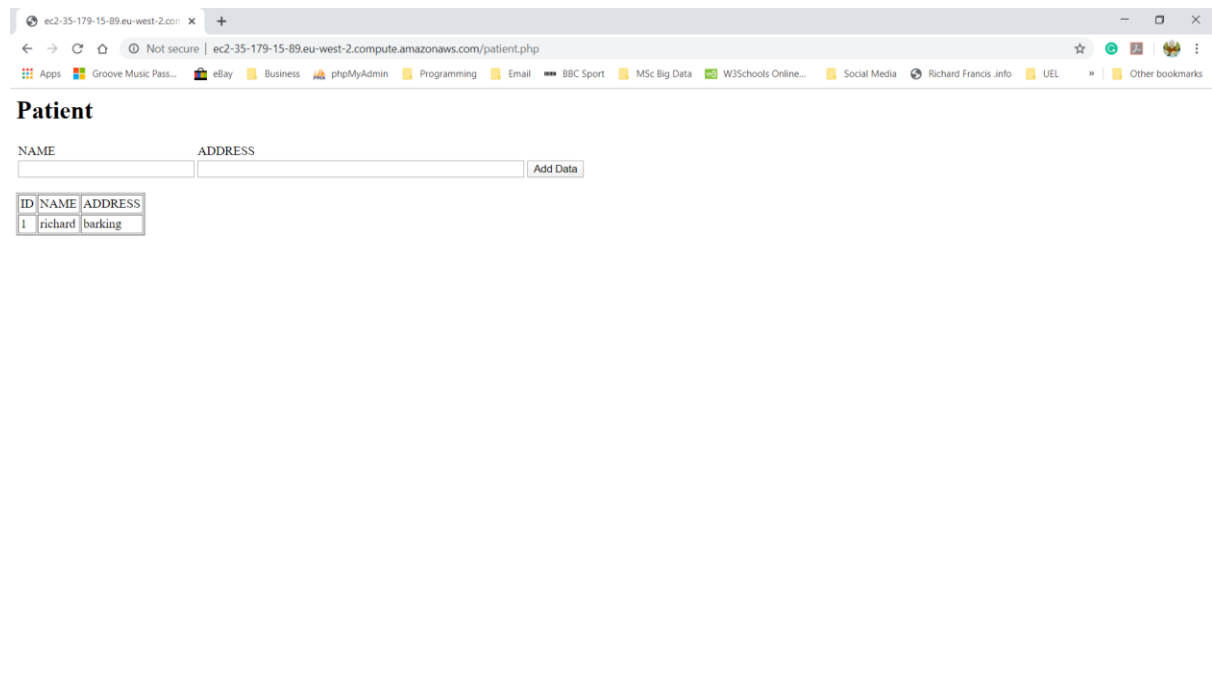
Auto Scaling and Application Load Balancer

```
Command Prompt
C:\Users\User>aws autoscaling describe-auto-scaling-groups --output text
AUTOSCALINGGROUPS
arn:aws:autoscaling:eu-west-2:280762879731:autoScalingGroup:9972a6a3-dda4-4e77-8fb2-ca51287a5751:autoScalingGroupName/gp-stack-WebServ
erGroup-5M8NHZYJAKCN gp-stack-WebServerGroup-5M8NHZYJAKCN 2020-03-25T15:54:10.136Z 300 2 0 EC2 gp-stack-LaunchConfig-PILUPVW
VZC6 5 1 False arn:aws:iam::280762879731:role/aws-service-role/autoscaling.amazonaws.com/ANSServiceRoleForAutoScaling subnet-b668bbfa,subnet
-ab9bdbc2,subnet-25b73b5f
AVAILABILITYZONES eu-west-2c
AVAILABILITYZONES eu-west-2a
AVAILABILITYZONES eu-west-2b
INSTANCES eu-west-2c Healthy i-07ac6963e87999693 t2.micro gp-stack-LaunchConfig-PILUPVWVZC6 InService False
INSTANCES eu-west-2b Healthy i-0b25542078c937de3 t2.micro gp-stack-LaunchConfig-PILUPVWVZC6 InService False
TAGS aws:cloudformation:logical-id True gp-stack-WebServerGroup-5M8NHZYJAKCN auto-scaling-group WebServerGroup
TAGS aws:cloudformation:stack-id True gp-stack-WebServerGroup-5M8NHZYJAKCN auto-scaling-group arn:aws:cloudformation:eu-west-2:280762879731:
stack/gp-stack/3c356ac0-6eae-11ea-ace0-060395953c4e
TAGS aws:cloudformation:stack-name True gp-stack-WebServerGroup-5M8NHZYJAKCN auto-scaling-group gp-stack
TARGETGROUPARNs arn:aws:elasticloadbalancing:eu-west-2:280762879731:targetgroup/gp-st-ALBta-198FQJA1PXH41/054d2408b99c75ec
TERMINATIONPOLICIES Default

C:\Users\User>aws elbv2 describe-load-balancers --output text
LOADBALANCERS ZHURV8PSTC4K8 2020-03-25T15:35:25.750Z gp-st-AppIi-1WTVMBAQLWLDQ-2144221443.eu-west-2.elb.amazonaws.com ipv4 arn:aws:elasti
cloudbalancing:eu-west-2:280762879731:loadbalancer/app/gp-st-AppIi-1WTVMBAQLWLDQ/c20346e9eb86f712 gp-st-AppIi-1WTVMBAQLWLDQ
ation vpc-32b7c95a
AVAILABILITYZONES subnet-25b73b5f eu-west-2a
AVAILABILITYZONES subnet-ab9bdbc2 eu-west-2c
AVAILABILITYZONES subnet-b668bbfa eu-west-2b
SECURITYGROUPS sg-828a1fe2
STATE active

C:\Users\User>
```


Enter data using a browser



The screenshot shows a web browser window with the address bar displaying `ec2-35-179-15-89.eu-west-2.compute.amazonaws.com/patient.php`. The page title is "Patient". Below the title, there are two input fields labeled "NAME" and "ADDRESS", followed by an "Add Data" button. Below these fields, there is a table with the following data:

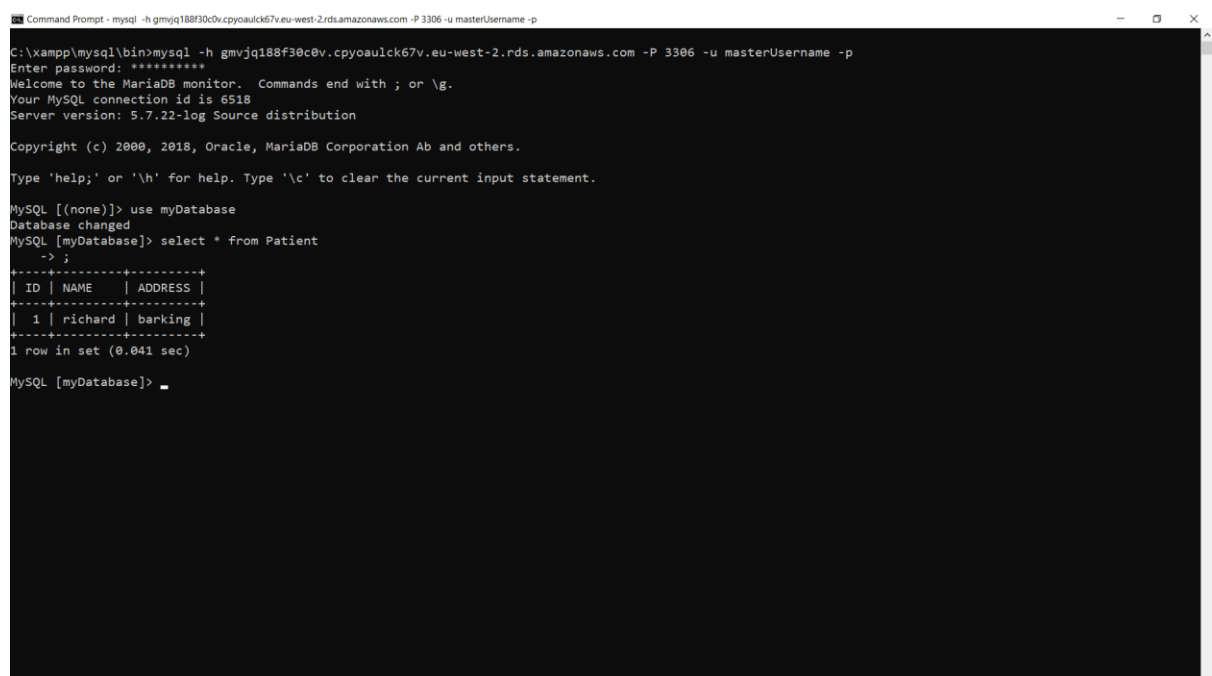
ID	NAME	ADDRESS
1	richard	barking

Once the stack has been created, simply configure, install and update.

Test the system works using a browser and enter data. The image below confirms that the data entered using a browser is the same as the user can see directly.

CloudFormation enable users to deploy applications much more efficiently. As an additional benefit the YAML syntax used by CloudFormation is very similar to English, so it acts as documentation for the system. An example of the YAML file is documented on the next page.

View the data using a direct connection



```
Command Prompt - mysql -h gmjq188f30c0v.cpyoaulck67v.eu-west-2.rds.amazonaws.com -P 3306 -u masterUsername -p
C:\xampp\mysql\bin>mysql -h gmjq188f30c0v.cpyoaulck67v.eu-west-2.rds.amazonaws.com -P 3306 -u masterUsername -p
Enter password: *****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 6518
Server version: 5.7.22-log Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> use myDatabase
Database changed
MySQL [myDatabase]> select * from Patient
-> ;
+----+-----+-----+
| ID | NAME  | ADDRESS |
+----+-----+-----+
| 1  | richard | barking |
+----+-----+-----+
1 row in set (0.041 sec)

MySQL [myDatabase]>
```

1.4.1 GP-STACK

Single line syntax used to deploy the stack

```
aws cloudformation deploy --template-file /CloudFormation/GP-STACK.yaml --stack-name gp-stack
```

GP-STACK. yaml

A copy of this template be downloaded [here](#)

AWSTemplateFormatVersion: 2010-09-09

Description: >-

AWS CloudFormation Template LAMP_Multi_AZ: Modified by Richard Francis for CN7026 - GP-VPC

Parameters:

VpcId:

Type: 'AWS::EC2::VPC::Id'

Description: VpcId of your existing Virtual Private Cloud (VPC)

ConstraintDescription: must be the VPC Id of an existing Virtual Private Cloud.

Subnets:

Type: 'List<AWS::EC2::Subnet::Id>'

Description: The list of SubnetIds in your Virtual Private Cloud (VPC)

ConstraintDescription: >-

must be a list of at least two existing subnets associated with at least two different availability zones. They should be residing in the selected Virtual Private Cloud.

KeyName:

Description: Name of an existing EC2 KeyPair to enable SSH access to the instances

Type: 'AWS::EC2::KeyPair::KeyName'

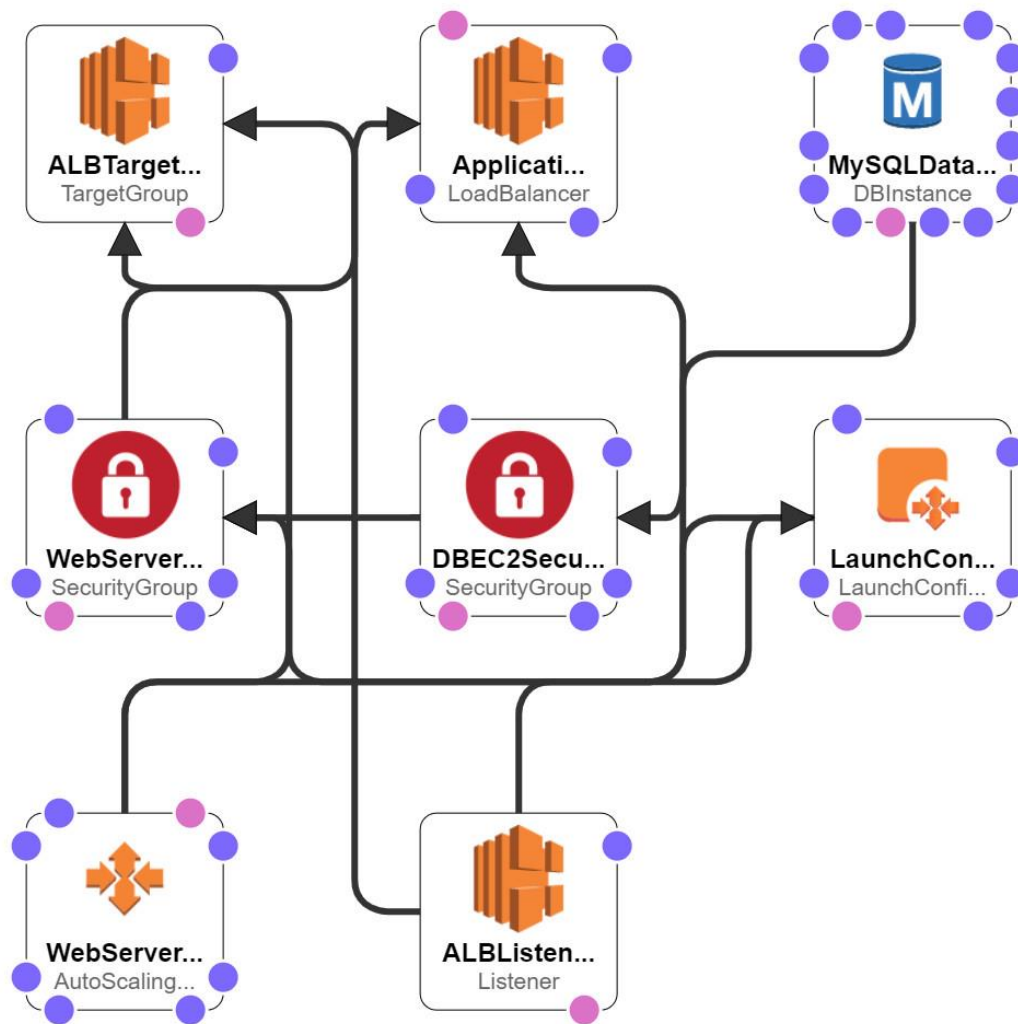
ConstraintDescription: must be the name of an existing EC2 KeyPair.

DBName:

Default: gpdb

Description: MySQL database name

1.4.2 GP-STACK.designer



1.4.3 GP-STACK-SET

Auto Scaling, Load Balancing template with SNS notifications using CloudWatch alarms

The screenshot displays the AWS CloudFormation console for the 'AutoScalingMultiAZWithNotifications' template. The top section shows a resource graph with the following components: ALBTargetGroup, Application Load Balancing, Launch Config, WebServer, AutoScalingGroup, and CloudWatch Alarms. The bottom section shows the template code in JSON format, which includes parameters for VPC, Subnets, and various resource configurations.

```
1 {
2   "AWSTemplateFormatVersion": "2010-09-09",
3   "Description": "AWS CloudFormation Sample Template AutoScalingMultiAZWithNotifications: Create a multi-az, load balanced and Auto Scaled sample web site running on an Apache Web Server. The application is configured to be highly available across multiple Availability Zones.",
4   "Parameters": {
5     "VPC": {
6       "Type": "AWS::EC2::VPC::Id",
7       "Description": "VPC of your existing Virtual Private Cloud (VPC)",
8       "ConstraintDescription": "Must be the VPC Id of an existing Virtual Private Cloud."
9     },
10    "Subnets": {
11      "Type": "List<AWS::EC2::Subnet::Id>",
12      "Description": "The list of Subnet Ids in your Virtual Private Cloud (VPC)",
13      "ConstraintDescription": "Must be a list of Subnet Ids in your Virtual Private Cloud (VPC)"
14    }
15  },
16  "Resources": {
17    "ALBTargetGroup": {
18      "Type": "AWS::ElasticLoadBalancingV2::TargetGroup",
19      "Properties": {
20        "VpcId": { "Ref": "VPC" },
21        "Subnets": { "Ref": "Subnets" },
22        "TargetType": "Instance",
23        "Port": 80
24      }
25    },
26    "ApplicationLoadBalancer": {
27      "Type": "AWS::ElasticLoadBalancingV2::LoadBalancer",
28      "Properties": {
29        "VpcId": { "Ref": "VPC" },
30        "Subnets": { "Ref": "Subnets" },
31        "Scheme": "internet-facing",
32        "LoadBalancerType": "Application"
33      }
34    },
35    "LaunchConfig": {
36      "Type": "AWS::AutoScaling::LaunchConfiguration",
37      "Properties": {
38        "ImageId": "ami-0c55b199",
39        "InstanceType": "t3.micro",
40        "SubnetId": { "Ref": "Subnets" },
41        "SecurityGroups": [ { "Ref": "SecurityGroup" } ],
42        "IamProfileName": "AWSAutoScaling",
43        "EBSOptimized": false,
44        "UserData": "curl -sSL https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip > awscli.zip; unzip awscli.zip; awscli --install"
45      }
46    },
47    "WebServer": {
48      "Type": "AWS::AutoScaling::AutoScalingGroup",
49      "Properties": {
50        "VpcId": { "Ref": "VPC" },
51        "Subnets": { "Ref": "Subnets" },
52        "LaunchConfigurationName": { "Ref": "LaunchConfig" },
53        "MinSize": 1,
54        "MaxSize": 3,
55        "DesiredCapacity": 2,
56        "HealthCheckType": "ELB",
57        "HealthCheckGracePeriod": 300,
58        "LoadBalancerNames": [ { "Ref": "ApplicationLoadBalancer" } ],
59        "NotificationConfigurations": [
60          {
61            "TopicARN": { "Ref": "Topic" },
62            "NotificationTypes": [ "LifecycleTransition" ]
63          }
64        ]
65      }
66    },
67    "Topic": {
68      "Type": "AWS::SNS::Topic",
69      "Properties": {
70        "Name": "AutoScalingMultiAZWithNotifications"
71      }
72    },
73    "Alarm": {
74      "Type": "AWS::CloudWatch::Alarm",
75      "Properties": {
76        "Name": "AutoScalingMultiAZWithNotifications",
77        "MetricName": "CPUUsage",
78        "Namespace": "AWS/EC2",
79        "Dimensions": [ { "Name": "InstanceId", "Value": { "Ref": "WebServer" } } ],
80        "Period": 300,
81        "EvaluationPeriods": 3,
82        "Threshold": 5,
83        "Statistic": "Average",
84        "AlarmActions": [ { "Ref": "Topic" } ]
85      }
86    }
87  },
88  "Outputs": {
89    "WebServer": {
90      "Value": { "Ref": "WebServer" }
91    }
92  }
93 }
```

[Download template](#)

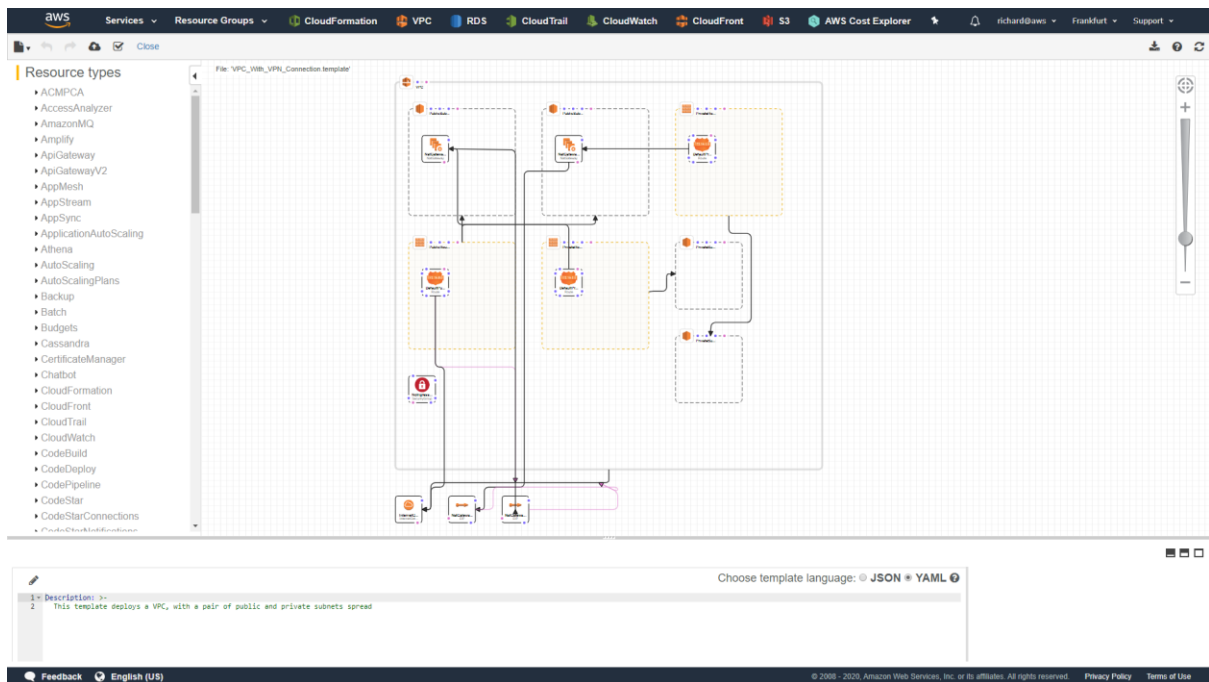
RDS MySQL with read replica

The screenshot displays the AWS CloudFormation console for the 'RDS_MySQL_With_Read_Replica' template. The top section shows a resource graph with the following components: RDS DB Instance, Read Replica, and CloudWatch Alarms. The bottom section shows the template code in JSON format, which includes parameters for DB Instance Name, DB Instance Class, DB Instance Size, and various resource configurations.

```
1 {
2   "AWSTemplateFormatVersion": "2010-09-09",
3   "Description": "AWS CloudFormation Sample Template RDS_MySQL_With_Read_Replica: Sample template showing how to create a highly-available, RDS DB Instance with a read replica. Modified by Richard Francis for CN7026",
4   "Parameters": {
5     "DBInstanceName": {
6       "Type": "String",
7       "Description": "The database name",
8       "MaxLength": 64,
9       "MinLength": 1,
10      "AllowedPattern": "[a-zA-Z][a-zA-Z0-9]*",
11      "ConstraintDescription": "Must begin with a letter and contain only alphanumeric characters."
12    },
13    "DBInstanceClass": {
14      "Type": "String",
15      "Description": "The database admin account username",
16      "MaxLength": 16,
17      "MinLength": 1,
18      "AllowedPattern": "[a-zA-Z][a-zA-Z0-9]*",
19      "ConstraintDescription": "Must begin with a letter and contain only alphanumeric characters."
20    },
21    "DBInstanceSize": {
22      "Type": "String",
23      "Description": "The database admin account password",
24      "MaxLength": 41,
25      "MinLength": 1,
26      "AllowedPattern": "[a-zA-Z0-9]*",
27      "ConstraintDescription": "Must contain only alphanumeric characters."
28    },
29    "DBInstanceStorage": {
30      "Type": "Number",
31      "Description": "The size of the database (GB)",
32      "MinValue": 5,
33      "MaxValue": 10240,
34      "ConstraintDescription": "Must be between 5 and 10240."
35    },
36    "DBInstanceClass": {
37      "Type": "String",
38      "Description": "The database instance type",
39      "Default": "db.m3.xlarge",
40      "AllowedValues": [
41        "db.m3.xlarge",
42        "db.m3.xlarge",
43        "db.m3.xlarge"
44      ]
45    }
46  },
47  "Resources": {
48    "DBInstance": {
49      "Type": "AWS::RDS::DBInstance",
50      "Properties": {
51        "DBInstanceName": { "Ref": "DBInstanceName" },
52        "DBInstanceClass": { "Ref": "DBInstanceClass" },
53        "DBInstanceSize": { "Ref": "DBInstanceSize" },
54        "DBInstanceStorage": { "Ref": "DBInstanceStorage" },
55        "Engine": "MySQL",
56        "EngineVersion": "5.7.26",
57        "InstanceProfileName": "AWSRDS",
58        "SecurityGroups": [ { "Ref": "SecurityGroup" } ],
59        "VpcId": { "Ref": "VPC" },
60        "SubnetGroup": { "Ref": "SubnetGroup" },
61        "AutoMinorVersionUpgrade": true,
62        "DeletionProtection": true,
63        "Monitoring": true,
64        "NotificationConfigurations": [
65          {
66            "TopicARN": { "Ref": "Topic" },
67            "NotificationTypes": [ "LifecycleTransition" ]
68          }
69        ]
70      }
71    },
72    "ReadReplica": {
73      "Type": "AWS::RDS::ReadReplica",
74      "Properties": {
75        "DBInstanceName": { "Ref": "DBInstanceName" },
76        "DBInstanceClass": { "Ref": "DBInstanceClass" },
77        "DBInstanceSize": { "Ref": "DBInstanceSize" },
78        "DBInstanceStorage": { "Ref": "DBInstanceStorage" },
79        "Engine": "MySQL",
80        "EngineVersion": "5.7.26",
81        "InstanceProfileName": "AWSRDS",
82        "SecurityGroups": [ { "Ref": "SecurityGroup" } ],
83        "VpcId": { "Ref": "VPC" },
84        "SubnetGroup": { "Ref": "SubnetGroup" },
85        "AutoMinorVersionUpgrade": true,
86        "DeletionProtection": true,
87        "Monitoring": true,
88        "NotificationConfigurations": [
89          {
90            "TopicARN": { "Ref": "Topic" },
91            "NotificationTypes": [ "LifecycleTransition" ]
92          }
93        ]
94      }
95    },
96    "Topic": {
97      "Type": "AWS::SNS::Topic",
98      "Properties": {
99        "Name": "RDS_MySQL_With_Read_Replica"
100      }
101    },
102    "Alarm": {
103      "Type": "AWS::CloudWatch::Alarm",
104      "Properties": {
105        "Name": "RDS_MySQL_With_Read_Replica",
106        "MetricName": "CPUUsage",
107        "Namespace": "AWS/RDS",
108        "Dimensions": [ { "Name": "DBInstanceIdentifier", "Value": { "Ref": "DBInstance" } } ],
109        "Period": 300,
110        "EvaluationPeriods": 3,
111        "Threshold": 5,
112        "Statistic": "Average",
113        "AlarmActions": [ { "Ref": "Topic" } ]
114      }
115    }
116  },
117  "Outputs": {
118    "DBInstance": {
119      "Value": { "Ref": "DBInstance" }
120    },
121    "ReadReplica": {
122      "Value": { "Ref": "ReadReplica" }
123    }
124  }
125 }
```

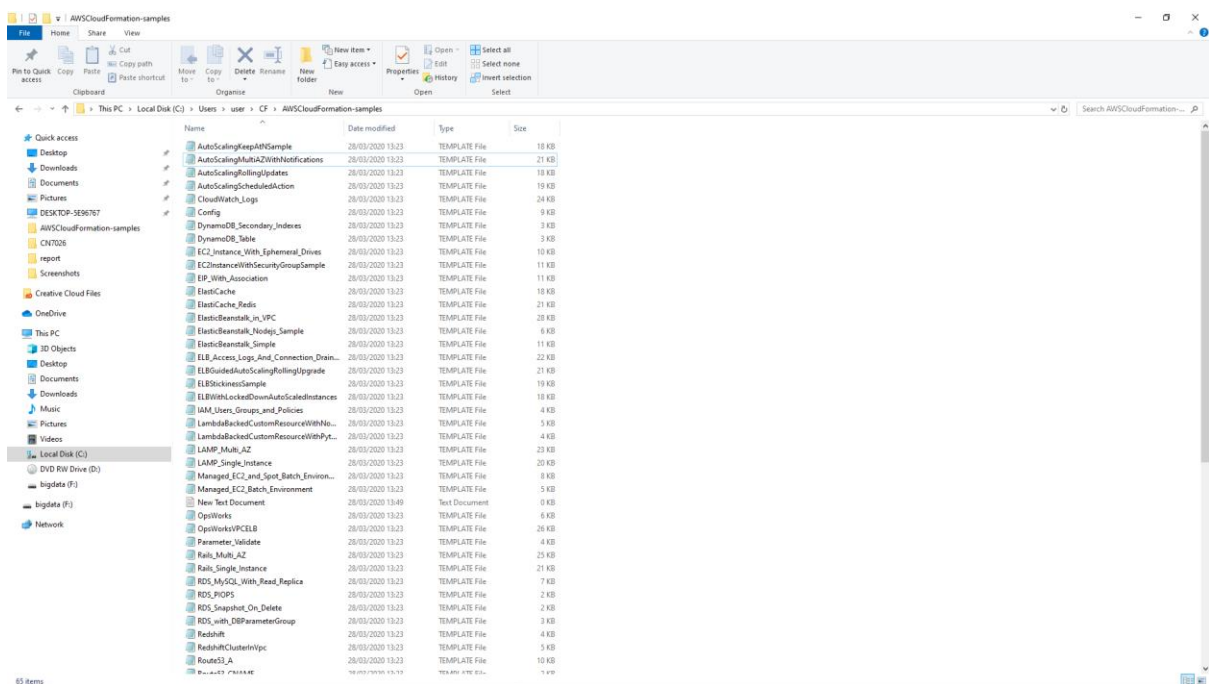
[Download template](#)

VPC with private and public subnets



[Download template](#)

Folder complete with CloudFormation template library



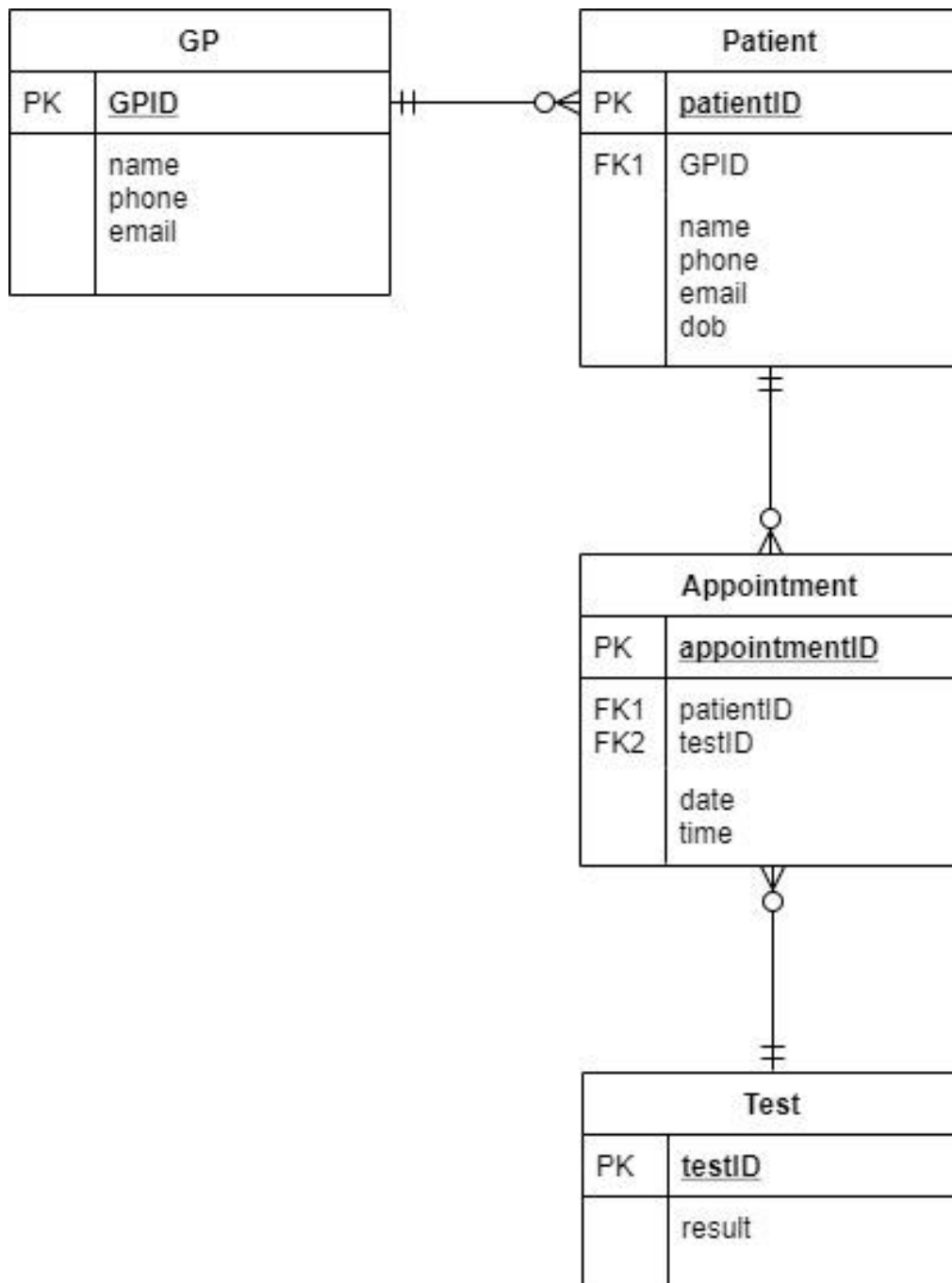
[Download folder](#)

Database Development

Appendix B

2.1 GP-ERD

The Entity Relationship Diagram (ERD) illustrated below represents the RDS database in the GP-VPC. The database tables include; a patient, GP, appointment, and test. The test table was designed to facilitate efficient coronavirus testing, using the RDS service in the GP-VPC.



2.2 GPDB.sql

```
CREATE DATABASE gpdb;
```

```
USE gpdb;
```

```
DROP TABLE IF EXISTS `GP`;
```

```
CREATE TABLE `GP` (  
  `GPID` int(11) NOT NULL,  
  `name` varchar(50) NOT NULL,  
  `phone` int(11) NOT NULL,  
  `email` varchar(255),  
  PRIMARY KEY (`GPID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `GP`(`GPID`,`name`,`phone`,`email`) VALUES
```

```
(7180,'dr david','7024451838','drdave@mail.com'),  
(7181,'dr peter','7022251822','drpeter@mail.com'),  
(7182,'dr gurav','7333551118','drgurav@mail.com'),  
(7183,'dr hawaad','7025341838','drhawaad@mail.com');
```

```
DROP TABLE IF EXISTS `patient`;
```

```
CREATE TABLE `patient` (  
  `patientID` int(11) NOT NULL,  
  `dob` date NOT NULL,  
  `name` varchar(50) NOT NULL,  
  `phone` int(11) DEFAULT NULL,  
  `email` varchar(255),  
  `GPID` int(11) DEFAULT NULL,  
  PRIMARY KEY (`patientID`),  
  KEY `GPID` (`GPID`),  
  CONSTRAINT `patient_ibfk_1` FOREIGN KEY (`GPID`) REFERENCES `gp` (`GPID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `patient`(`patientID`,`name`,`dob`,`phone`,`email`,`GPID`) VALUES
```

```
(1012,'Jean', '2000-01-02', '7025551838','jean@mail.com','7180'),  
(1013,'Susie', '1990-03-22','7025551838','sue@mail.com','7182'),  
(1014,'Richard', '1967-08-21','7025551838','rich@mail.com','7181'),  
(1015,'Paul', '2001-01-02', '7025551838','paul@mail.com','7182'),  
(1016,'Gary', '1970-11-05', '7025551838','gary@mail.com','7183'),  
(1017,'Andrew', '1959-06-06', '7025551838','asf@mail.com','7180');
```

```
CREATE TABLE `appointment` (  
  `appointmentID` int(11) NOT NULL,  
  `appointmentDate` datetime NOT NULL,  
  `testID` int(11) DEFAULT NULL,  
  `patientID` int(11) DEFAULT NULL,
```



```
PRIMARY KEY (`appointmentID`),  
KEY `testID` (`testID`),  
KEY `patient` (`patientID`),  
CONSTRAINT `test_ibfk_1` FOREIGN KEY (`testID`) REFERENCES `test` (`testID`),  
CONSTRAINT `patient_ibfk_2` FOREIGN KEY (`patientID`) REFERENCES `patient` (`patientID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO appointment (appointmentID, appointmentDate, testID, patientID) VALUES  
(9003, '2020-02-05 14:29:36', 30012, 1013),  
(9004, '2020-02-05 15:22:26', 30013, 1016),  
(9005, '2020-02-06 11:49:38', 30014, 1014);
```

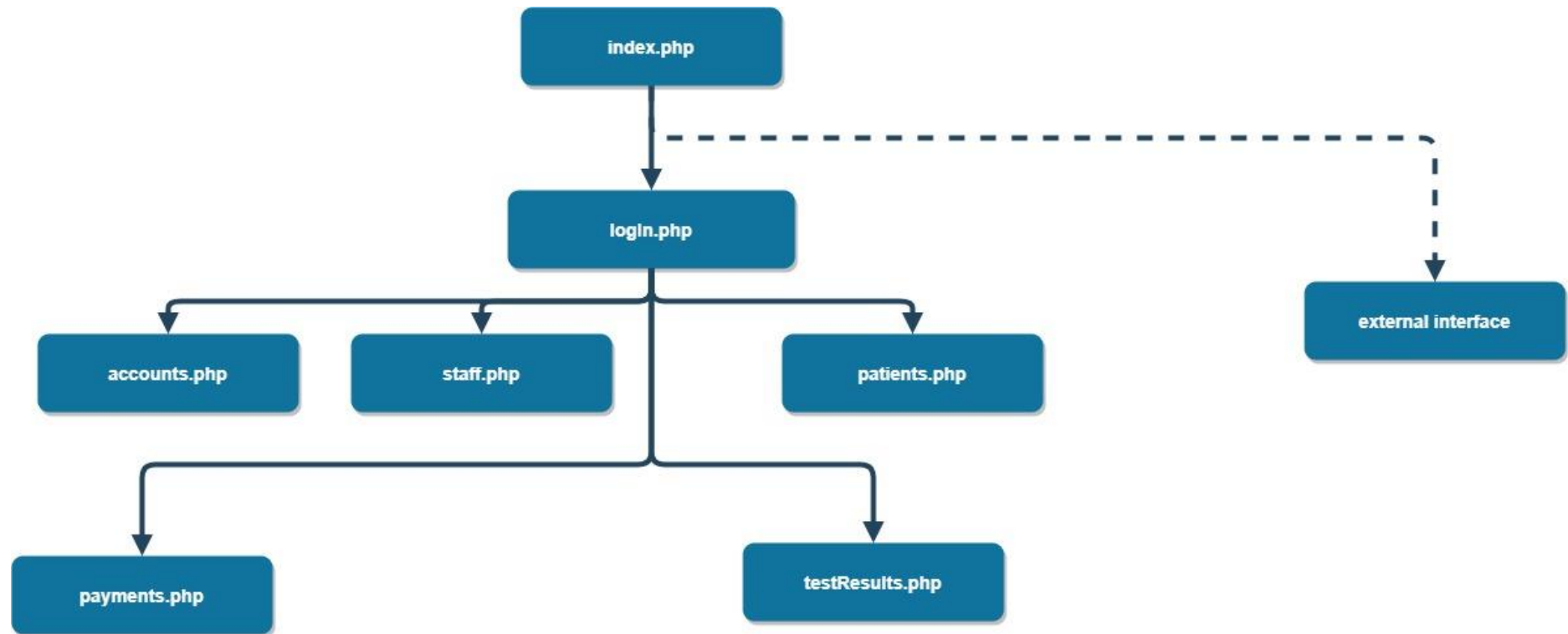
```
CREATE TABLE `test` (  
  `testID` int(11) NOT NULL,  
  `result` BOOLEAN NOT NULL,  
  PRIMARY KEY (`testID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO test (testID, result) VALUES  
(30012,true),  
(30013,true),  
(30014,false);
```

Website Development

Appendix C

3.1 Website Architecture



3.2 Patient.php

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Patient</h1>
<?php

/* Connect to MySQL and select the database. */
$connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " . mysqli_connect_error();

$database = mysqli_select_db($connection, DB_DATABASE);

/* Ensure that the Patient table exists. */
VerifyPatientTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the Patient table. */
$patient_name = htmlentities($_POST['NAME']);
$patient_address = htmlentities($_POST['ADDRESS']);

if (strlen($patient_name) || strlen($patient_address)) {
    Addpatient($connection, $patient_name, $patient_address);
}
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
<table border="0">
<tr>
<td>NAME</td>
<td>ADDRESS</td>
</tr>
<tr>
<td>
<input type="text" name="NAME" maxlength="45" size="30" />
</td>
<td>
<input type="text" name="ADDRESS" maxlength="90" size="60" />
</td>
<td>
<input type="submit" value="Add Data" />
</td>
</tr>
</table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
<tr>
```

```

        <td>ID</td>
        <td>NAME</td>
        <td>ADDRESS</td>
    </tr>

```

```
<?php
```

```
$result = mysqli_query($connection, "SELECT * FROM Patient");
```

```

while($query_data = mysqli_fetch_row($result)) {
    echo "<tr>";
    echo "<td>",$query_data[0], "</td>",
        "<td>",$query_data[1], "</td>",
        "<td>",$query_data[2], "</td>";
    echo "</tr>";
}
?>

```

```
</table>
```

```
<!-- Clean up. -->
```

```
<?php
```

```

    mysqli_free_result($result);
    mysqli_close($connection);

```

```
?>
```

```
</body>
```

```
</html>
```

```
<?php
```

```
/* Add an patient to the table. */
```

```

function Addpatient($connection, $name, $address) {
    $n = mysqli_real_escape_string($connection, $name);
    $a = mysqli_real_escape_string($connection, $address);

```

```
    $query = "INSERT INTO Patient (NAME, ADDRESS) VALUES ('$n', '$a');";
```

```

    if(!mysqli_query($connection, $query)) echo("<p>Error adding patient data.</p>");
}

```

```
/* Check whether the table exists and, if not, create it. */
```

```

function VerifyPatientTable($connection, $dbName) {
    if(!TableExists("Patient", $connection, $dbName))
    {
        $query = "CREATE TABLE Patient (
            ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
            NAME VARCHAR(45),

```

```

        ADDRESS VARCHAR(90)
    );

    if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</p>");
}
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
    $t = mysqli_real_escape_string($connection, $tableName);
    $d = mysqli_real_escape_string($connection, $dbName);

    $checktable = mysqli_query($connection,
        "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t' AND
        TABLE_SCHEMA = '$d'");

    if(mysqli_num_rows($checktable) > 0) return true;

    return false;
}
?>

```

Project Management

Appendix D

First meeting			
20.02.2020		15:00	ITC04
Meeting called by	Richard		
Type of meeting	Group consolidation		
Facilitator	Rana		
Note taker	Sony		
Timekeeper	Sony		
Attendees	Sony, Richard, Rana, Supun		
Agenda Topic 1			
1 hour	Richard		
Discussion	Progress report		
Conclusions	Richard demonstrated IAM, SES, CLI, and Elastic Beanstalk. Next week Sony and Nas will discuss progress on disaster recovery or virtualization		
Agenda Topic 2			
10 minutes	Sony		
Discussion	Project management		
Conclusions	Sony will take more responsibility. Richard will take less		
		Person	Deadline
Software development – Java/C#		Supun	26.03.20
Agenda Topic 3			
1 hour	Rana		
Discussion	What services are we likely to use		
Conclusions	EC2, lambda, S3 and Elastic Beanstalk		
Task allocation		Person	Deadline
Alexa, S3, RDBMS		Rana	27.02.20
AWS CLI, SES, IAM		Richard	27.02.20
AWS SDK for Java, Lambda		Supun	27.02.20
Disaster recovery and Virtualization		Sony	27.02.20

Second meeting			
27.02.2020		15:00	ITC04
Meeting called by	Richard		
Type of meeting	Group progress		
Facilitator	Rana		
Note taker	Sony		
Timekeeper	Sony		
Attendees	Sony, Richard, Rana, Supun		
Agenda Topic 1			
1 hour	Richard		
Discussion	Progress report		
Conclusions	<p>Sony discussed virtualization and explained how it could be used in our project.</p> <p>Rana discussed EC2 and gave examples of how it could be used in our project.</p> <p>Richard discussed the 5 pillars of a well architecture framework, and shared examples of the research we need to complete.</p>		
Agenda Topic 2			
30 minutes	Rana		
Discussion	Database development and documentation		
Conclusions	Richard shared examples of previous projects, Rana and Supun agreed to base their work on the 5 pillars.		
Agenda Topic 3			
1 hour	Sony		
Discussion	Virtualization		
Conclusions	The group all understand the benefits		
Task allocation		Person	Deadline
Alexa, S3, RDBMS, Performance pillar		Rana	05.03.20
AWS Lambda, S3, RDBMS, Security pillar		Richard	05.03.20
Node JS, Operational excellence pillar		Supun	05.03.20
Disaster Recovery, Reliability pillar		Sony	05.03.20

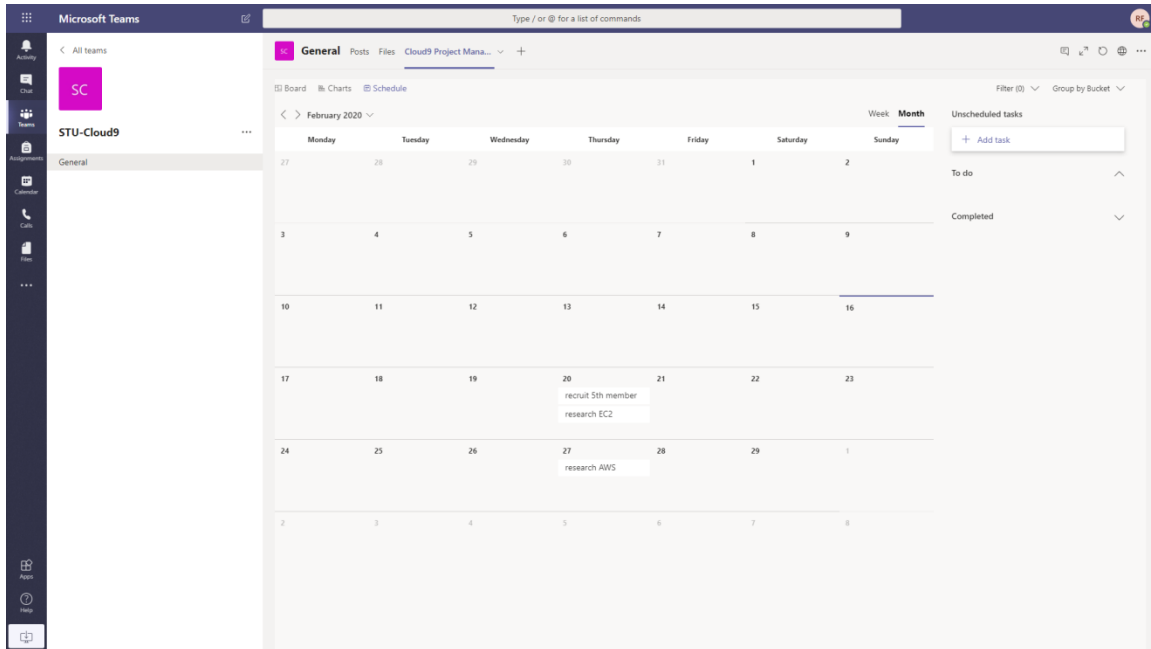
Progress meeting			
05.03.2020		15:00	ITC04
Meeting called by	Sony		
Type of meeting	Group progress		
Facilitator	Rana		
Note taker	Richard		
Timekeeper	Supun		
Attendees	Sony, Richard, Rana, Supun		
Agenda Topic 1			
30 minutes	Richard		
Discussion	GP Cloud Migration, Data Flow and Use Case Examples		
Conclusions	Richard created a vpc-rds-ec2 system and demonstrated updating the database and web pages. Complete GP requirements analysis		
Agenda Topic 2			
30 minutes	Rana and Supun		
Discussion	Database development and documentation		
Conclusions	Complete GP and DB requirements analysis		
Agenda Topic 3			
30 minutes	Sony		
Discussion	draw.io		
Conclusions	use draw.io and update the reliability.doc – use own words		
Task allocation		Person	Deadline
DB Development, Performance, Cost		Rana	12.03.20
DynamoDB, Security, Lambda, Kinesis, Cost, CloudFront, System Architecture		Richard	12.03.20
Node JS, Operations, DB Development, Cost		Supun	12.03.20
Reliability, draw.io, Cost		Sony	12.03.20

Progress meeting			
12.03.2020		15:00	ITC04
Meeting called by	Rana		
Type of meeting	Group progress		
Facilitator	Rana		
Note taker	Richard		
Timekeeper	Richard		
Attendees	Richard, Rana		
Agenda Topic 1			
30 minutes	Richard		
Discussion	Progress Report		
Conclusions	Richard modelled the vpc and documented the development process. Rana and Supun modelled the database and shared with the group. Sony sent a template GP requirements analysis is incomplete		
Agenda Topic 2			
30 minutes	Rana and Supun		
Discussion	Database development and documentation		
Conclusions	GP and DB requirements analysis are incomplete		
Agenda Topic 3			
30 minutes	Richard		
Discussion	reliability		
Conclusions	update the reliability.doc – use own words		
Task allocation		Person	Deadline
DB Development, Performance		Rana	19.03.20
RDS, Security, CloudFront, CloudTrail, System Architecture, IAM, System Documentation		Richard	19.03.20
Node JS, Operations, DB Development		Supun	19.03.20
Reliability		Sony	19.03.20

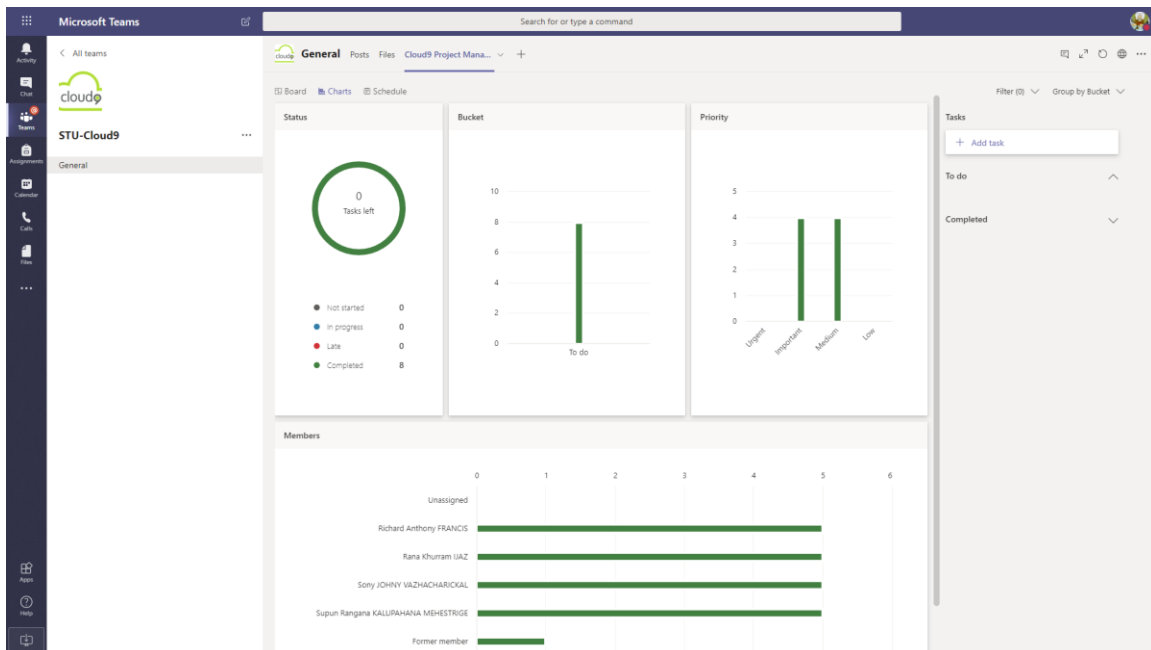
Project Management Documentation

Due to the virus outbreak group work was suspended. However, it is important to note that the schedule was kept, and tasks were completed on time.

Project schedule



Project progress



Miscellaneous

Appendix E

5.1 CloudTrail Logs

The logs highlighted below are a real life example. Using these logs administrators and business owners can audit, analyse and decide what future actions or processes are required to improve the system.

Log file example

```
85446540137b3a81572684bbf84a43d53837bf7ae4f788b474c64913127ac5b5 research-aws
[09/Mar/2020:03:24:40 +0000] 188.31.92.221
85446540137b3a81572684bbf84a43d53837bf7ae4f788b474c64913127ac5b5 3F5B8224B83A2CAE
REST.GET.ACL - "GET /?acl HTTP/1.1" 200 - 480 - 5 - "-" "S3Console/0.4, aws-internal/3 aws-sdk-
java/1.11.719 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 OpenJDK_64-Bit_Server_VM/25.242-
b08 java/1.8.0_242 vendor/Oracle_Corporation" -
4EXxoHL/D1Cf0hp96y9HNZNFK47wbbOqNVb8I0bHOZYO2dP/xGpMVP/L/ruIjgqvfoKN3Tvk3so= SigV4
ECDHE-RSA-AES128-SHA AuthHeader research-aws.s3.eu-west-2.amazonaws.com TLSv1.2
```

5.2 IAM.json

Using json type files IAM administrators and business owners can restrict access to AWS resources. Temporary access may be provided; roles and groups are also used.

JSON file example

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:ACCOUNT-ID-WITHOUT-
HYPHENS:table/${aws:username}"
  }
}
```

5.3 Micro Services

Application Load Balancer:	Integrated with CloudWatch to ensure traffic is directed to healthy instances.
Auto Scaling Group:	Adjusting capacity to meet demand and provide a layer of security using security groups.
NAT Gateway:	You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances. A NAT gateway can support up to 55,000 simultaneous connections to each unique destination.
Internet Gateway:	A horizontally scaled redundant and highly available VPC component that allows communication between instances in the VPC and the internet
Subnets:	Private and public range of IP addresses in the GP-VPC
Route Table:	Specify how network traffic is directed from within the VPC.
Network Interface:	<p>Gives the web server a public IP address</p> <ul style="list-style-type: none">• <u>Elastic IP Addresses:</u> An <i>Elastic IP address</i> is a static, public IPv4 address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface for any VPC in your account.
Security Groups:	Control inbound and outbound traffic for the instances
Network ACL's	Control inbound and specify deny for outbound traffic for your subnets. Must be subnet level.
Flow Logs:	Capture the traffic that flows to and from the network interfaces.
Endpoints	A VPC endpoint enables you to privately connect your VPC to supported AWS services.

5.4 Resources

The following lists the services used and tutorials completed while creating the GP-VPC environment. This list does not include resources highlighted in the report.

Walkthrough: Use AWS CloudFormation Designer to Create a Basic Web Server

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/working-with-templates-cfn-designer-walkthrough-createbasicwebserver.html>

Walkthrough: Create a Scalable, Load-balancing Web Server

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/example-templates-autoscaling.html>

Tutorial: Create a Web Server and an Amazon RDS Database

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/TUT_WebAppWithRDS.html

Deploying a high-availability PHP application with an external Amazon RDS database to EB

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/php-ha-tutorial.html>

Configuring a Static Website Using a Custom Domain Registered with Route 53

<https://docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html#add-bucket-policy-root-domain>

CloudFront Documentation: Configuring Secure Access and Restricting Access to Content

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/SecurityAndPrivateContent.html>

Tutorial: Schedule AWS Lambda Functions Using CloudWatch Events

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html>

CloudWatch Documentation: Create Alarms to Stop, Terminate, Reboot, or Recover an Instance

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/UsingAlarmActions.html#AddingStopActions>

Build a Serverless Web Application with AWS Lambda, Amazon API Gateway, Amazon S3, Amazon DynamoDB, and Amazon Cognito

<https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/>

Example: Create an IPv4 VPC and Subnets Using the AWS CLI

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-subnets-commands-example.html>

Tutorial: Use CodeDeploy to Deploy an Application to an Amazon EC2 Auto Scaling Group

<https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-auto-scaling-group.html>

Using AWS Lambda with Amazon S3 Events

<https://docs.aws.amazon.com/lambda/latest/dg/with-s3.html>

IAM Best Practices

<https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html>

Automated Response and Remediation with AWS Security Hub

<https://aws.amazon.com/blogs/security/automated-response-and-remediation-with-aws-security-hub/>

Create a Network File System with Amazon Elastic File System (EFS)

https://aws.amazon.com/getting-started/tutorials/create-network-file-system/?trk=gs_card

Send an Email with Amazon SES

<https://aws.amazon.com/getting-started/hands-on/send-an-email/>

Tutorial: Launch and configure a LAMP instance in Amazon Lightsail

https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-tutorial-launching-and-configuring-lamp

Well Architected - Learn, measure, and build using architectural best practices

<https://aws.amazon.com/architecture/well-architected/>

Internet Traffic Privacy in Amazon VPC

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html

Security Best Practices for Your VPC

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html>

Get Started with Amazon SageMaker Notebook Instances and SDKs

<https://docs.aws.amazon.com/sagemaker/latest/dg/gs-console.html>

Store, Protect, Optimize Your Healthcare Data with AWS: Part 1

<https://aws.amazon.com/blogs/architecture/store-protect-optimize-your-healthcare-data-with-aws/>

Store, Protect, Optimize Your Healthcare Data with AWS: Part 2

<https://aws.amazon.com/blogs/architecture/store-protect-optimize-your-healthcare-data-with-aws-part-2/>

Healthcare & Life Sciences

<https://aws.amazon.com/health/>

CSSE COVID-19 Dataset

https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data

Tutorial: Using AWS Lambda with Amazon Kinesis

<https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis-example.html>

Resize an Image in AWS S3 Using a Lambda Function

<https://levelup.gitconnected.com/resize-an-image-in-aws-s3-using-lambda-function-dc386afd4128>

